НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

До захисту допущено:
Завідувач кафедри
_____ Сергій СТІРЕНКО
«___»_____2020р.

Дипломний проект

на здобуття ступеня бакалавра
за освітньо-професійною програмою «Комп'ютерні системі та мережі»
спеціальності 123 «Комп'ютерна інженерія»
на тему: «Файловий менеджер в інтернет-середовищі»

Виконав:
студент IV курсу, групи ІО-62
Палієнко Артем Павлович _____

Керівник:
Доцент кафедри ОТ, к.т.н.,
Антонюк Андрій Іванович _____

Консультант з нормоконтролю
Професор кафедри ОТ, д.т.н.,
Сімоненко Валерій Павлович _____

Рецензент:
Доцент кафедри ТК, к.т.н.,
Пасько Віктор Петрович _____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ - 2020 року

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського’’
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 123 «Комп’ютерна інженерія»

Освітньо-професійна програма – «Комп’ютерні системі та мережі»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Сергій СТІРЕНКО
«___»_____2020р.

ЗАВДАННЯ
на дипломний проект студенту
Палієнку Артему Павловичу

1. Тема проекту «Файловий менеджер в інтернет-середовищі», керівник проекту Антонюк Андрій Іванович, доцент, к.т.н., с.н.с., затверджені наказом по університету від « 07 » травня 2020 року № 1081-с

2. Термін подання студентом проекту _____

3. Вихідні дані до проекту
   - Мова програмування Python
   - Фреймворк Django
   - Мова розмітки HTML та каскадні стилі CSS

4. Зміст роботи
   - Аналіз предметної області, огляд існуючих рішень
   - Аналіз інструментів та засобів для реалізації
   - Створення відповідного програмного продукту
   - Огляд отриманого результату

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо)
   - Структура та зв’язок між вузлами розробленого файлового менеджера
   - Взаємодія модулів програми
   - Алгоритм обробки запитів

6. Консультанти розділів проекту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
нормоконтроль	проф. Сімоненко В. П.		

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів проекту	Примітки
1	Затвердження теми проекту	01.09.2019	
2	Вивчення та аналіз завдання	18.01.2020	
3	Аналіз існуючих рішень	04.02.2020	
4	Вибір засобів розробки	11.02.2020	
5	Розробка програмного продукту	08.03.2020	
6	Тестування	10.04.2020	
7	Оформлення пояснювальної записки	01.05.2020	
8	Передзахист	26.05.2020	
9	Захист	18.06.2020	

Студент Артем ПАЛІЄНКО

Керівник Андрій АНТОНЮК

Анотація

В цій бакалаврській дипломній роботі реалізовано веб-сервіс, який клієнт може використовувати у якості файлового менеджера для свого персонального комп’ютера або ноутбука. Для досягнення мети були використані інтегроване середовище розробки JetBrains PyCharm та фреймворк Django.

У роботі обґрунтовано вибір технологій розробки, доведена актуальність створеного продукту та детально описано принцип його роботи.

Загальний обсяг дипломного проекту: опис альбому(1 арк.), технічне завдання(3 арк.), пояснювальна записка(56 арк.), додатки(18 арк.).

Ключові слова: файловий менеджер, інформаційні технології, веб-сервіс, Python, HTML, Django.

Annotation

This work for a Bachelor's Degree implemented a web-service, that a client can use as a file manager for their personal computer or laptop. The JetBrains PyCharm development environment and Django framework were used to achieve this goals.

The choice of development technologies is substantiated in the work, the relevance of the created product is proved, and the principle of its work is described in detail.

The total volume of the diploma project: album description (1 sheet), technical assignment (3 sheets), explanatory note (56 sheets), applications (18 sheets).

Keywords: file manager, information technologies, web-service, Python, HTML, Django.

Опис альбому

№ з/п	Формат	Позначення	Найменування	Кількість аркушів	Примітка
			Документація загальна		
			Заново розроблена		
1	А4	ІАЛЦ.045440.002 ТЗ	Файловий менеджер в	3	
			інтернет-середовищі		
			Технічне завдання		
2	А4	ІАЛЦ.045440.003 ПЗ	Файловий менеджер в	56	
			інтернет-середовищі		
			Пояснювальна записка		
3	А3	ІАЛЦ.045440.004 Д1	Файловий менеджер в	1	
			інтернет-середовищі		
			Структурна схема		
4	А3	ІАЛЦ.045440.005 Д2	Файловий менеджер в	1	
			інтернет-середовищі		
			Функціональна схема		
5	А3	ІАЛЦ.045440.006 Д3	Файловий менеджер в	1	
			інтернет-середовищі		
			Принципова схема		
6	А4	ІАЛЦ.045440.007 Д4	Файловий менеджер в	11	
			інтернет-середовищі		
			Ключові елементи коду програми		

Технічне завдання

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

Дане технічне завдання розповсюджується на розробку програмного забезпечення для керування файлами комп’ютера з інтернет-середовища. Областю застосування такого продукту є інтернет та хмарні технології, їх майбутні можливості використання.

2. ПІДСТАВИ ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на виконання роботи кваліфікаційно-освітнього рівня «бакалавр комп’ютерної інженерії», затверджене кафедрою обчислювальної техніки Національного технічного університету України «Київський політехнічний інститут ім. І. Сікорського».

3. МЕТА ТА ПРИЗНАЧЕННЯ РОЗРОБКИ

Метою розробки є створення зручного файлового менеджера, який працюватиме в інтернет-середовищі та легко і точно виконуватиме звичні для подібних програм операції. Призначенням цього продукту є допомога як звичайним так і професійним користувачам швидко працювати з файлами.

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелами, що використовуються під час розробки, є офіційна документація до засобів, використаних для створення програмного продукту, поява останнім часом значної кількості публікацій та літератури в Інтернеті стосовно даної теми.

5. ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до розробляємого продукту

- Надання можливості користування усіма стандартними функціями файлового менеджера
- Коректність, швидкість та точність виконання операцій клієнта
- Зручний інтерфейс

5.2. Вимоги до програмного забезпечення

- Python версії 3.5 и вище(сервер)
- Середовище розробки для мови Python(сервер)
- Фреймворк Django версії 2.2 и вище(сервер)
- Будь-який зручний веб-браузер(клієнт)

5.3. Вимоги до апаратної частини

- Підключення до Інтернету(клієнт, сервер)

6. ЕТАПИ РОЗРОБКИ

Етап	Дата
Вивчення та аналіз завдання	18.01.2020
Аналіз існуючих рішень	04.02.2020
Вибір засобів розробки	11.02.2020
Розробка програмного продукту	08.03.2020
Тестування	23.04.2020
Оформлення пояснювальної записки	01.05.2020

Пояснювальна записка

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

API – Application Program Interface

COM – Communication port

CRM – Customer Relationship Management

CRUD – Create/Read/Update/Delete

CSS – Cascading Style Sheets

DLL – Dynamic Link Library

DNS – Domain Name System

FTP – File Transfer Protocol

HTML – HyperText Markup Language

HTTP – HyperText Transfer Protocol

IDC – International Data Corporation

IP – Internet Protocol

IT – Information Technology

LDAP – Lightweight Directory Access Protocol

LLVM – Low-Level Virtual Machine

MTV – Model-Template-View

MVC – Model-View-Controller

NFS – Network File System

ORM – Object-Relational Mapping

OSI – Open System Interconnection

RPC – Remote Procedure Call

RSS – Really Simple Syndication

SOAP – Simple Object Access Protocol

SQL – Standartized Query Language

SSH – Secure SHell

SSL – Secure Sockets Layer

SWIG – Simple Wrapper and Interface Generator

TCP – Transmission Control Protocol

TIOBE – company ‘‘The Importance Of Being Earnest’’

TV – Television

URL – Uniform Resource Locator

USB – Universal Serial Bus

VMS – Virtual Memory System

WSGI – Web Server Gateway Interface

XML – eXtensible Markup Language

Б – байт; 8 біт

БД – база даних

ВМ – віртуальна машина

ГБ – гігабайт; 1000000000 або 1073741824 (залежно від контексту) байти

КБ – кілобайт; 1000 або 1024 (залежно від контексту) байти

МБ – мегабайт; 1000000 або 1048576 (залежно від контексту) байт

ООП – об’єктно-орієнтоване програмування

ОС – операційна система

ПК – персональний комп’ютер

ВСТУП

Уявити в наш час житло людини без комп'ютера майже неможливо. Ви не знайдете фірму, офіс, завод чи банк, в яких не застовуються подібні пристрої. Люди використовують персональні комп'ютери для збереження необхідної їм інформації, даних та файлів. Але якими б корисними не були ПК, з кожним днем все більшої необхідності та популярності набувають технології, які дозволять уникнути значної кількості дискомфорту спричиненого користуванням комп'ютерами. Великі системні блоки, необхідність в значному об'ємі пам'яті та спеціальному робочому місці, значна кількість шнурів, випадки поломки – це проблеми, які необхідно вирішити якнайшвидше для продовження технологічного прогресу людства.

У цій пояснювальній записці детально описуються можливості інтернет-середовища на даний момент, процес створення та дослідження програмного продукту. Останній буде мати вигляд сайту та виконуватиме роль файлового менеджеру, надаючи нам змогу в повній мірі керувати файлами, які знаходяться на віддаленому сервері. Такий механізм надасть змогу клієнту отримати доступ до своїх даних, маючи лише веб-браузер на компактному та зручному для нього пристрої. Гарним прикладом для цього буде використання набираючої популярність технології Smart TV(телевізор з можливістю доступу до Інтернету). Комбінація цієї технології та створеного в контексті дипломної роботи програмного продукту надасть змогу подібних до Smart TV пристроїв перейняти ще більше функцій персонального комп'ютера, а з часом і зовсім замінити їх. Усі ж файли користувача можуть віддалено зберігатися на кластерах та серверах великих компаній(наприклад Google), в найкращому випадку це надасть даним клієнта надійний захист, відсутність необхідних дій у випадках виходу з ладу окрім повідомлення компанії, значний комфорт та найвища якість обладнання.

РОЗДІЛ 1

АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Загальні відомості та переваги сфери

Інтернет-середовище – всесвітня система поєднаних між собою комп'ютерних мереж, що базуються на комплекті Інтернет-протоколів. [14] Інтернет ще називають мережею мереж, бо складається він із мільярдів локальних та глобальних, приватних та публічних мереж, пов'язаних між собою використанням різних дротових, оптичних і бездротових технологій(рис. 1.1).

Рис. 1.1 Абстрактне представлення можливостей Інтернету

Інтернет не має централізованого управління, правил користування чи доступу. Лише завдяки протоколу IP (Internet Protocol) та принципу маршрутизації пакетів даних стало можливим об'єднання різних за архітектурою мереж. Протокол IP був спеціально створений таким чином, щоб не зважати на фізичні канали зв'язку. Отже будь-яка система передачі цифрових даних може надсилати інтернет-трафік. На з'єднанні мереж спеціально створені маршрутизатори виконують сортування та перенаправлення усіх пакетів даних, в залежності від IP-адрес одержувачів цих пакетів. Таким чином протокол IP

формує єдиний всеохоплюючий адресний простір, проте в кожній окремій мережі також може бути створений свій власний адресний підпростір. Така організація IP-адрес надає можливість маршрутизаторам однозначно встановити подальший напрям для кожного, навіть малого, пакету даних. В результаті між різними мережами Інтернету не виникають конфлікти а інформація безпомилково передається по всій планеті.

В такому випадку протокол є способом взаємодії та обміну даними між пристроями під час роботи в мережі. Щоб різні комп'ютери мали можливість працювати разом, вони повинні використовувати однакові протоколи. Сукупність усіх протоколів має назву ‘‘стек протоколів TCP/IP’’. Найпоширеніші мережеві протоколи відповідно до моделі OSI наступні.

Прикладний рівень: HTTP, HTTPS, SSH, FTP, IMAP, LDAP, POP3, DNS, SMTP, XMPP, Telnet.

Сеансовий рівень: SSL, TLS.

Транспортний рівень: TCP, UDP.

Мережевий рівень: IP, RIP, BGP, IS-IS, IGMP, ICMP, OSPF, EIGRP.

Канальний рівень: SLIP, Ethernet, HDLC, PPP, Frame relay.

Крім того існує ще цілий ряд нестандартизованих, але вже дуже популярних протоколів. Зазвичай, це є протоколи децентралізованого обміну файлами і текстовими повідомленнями; на багатьох з них навіть побудовані цілі файлообмінні мережі. Наприклад: OSCAR, BitTorrent, Gnutella, Skype, CDDB, eDonkey.

Переваги використання Інтернету:

1. Комунікація

Головною метою мережі Інтернет завжди був зв'язок. Інтернет досяг значного успіху в цій галузі, більш того, новітні технології розширюють можливості зв'язку, роблячи його швидшим та надійнішим. З появою Інтернету, спілкування з будь-яким куточком Землі стало доступним і простим, як розмова з сусідом. Існує безліч послуг, за допомогою яких ви легко можете налагодити

дружбу з людиною, поділитися своїми думками, вивчати культуру різних національностей. [13]

2. Інформація

Інформація, ймовірно, найбільша перевага, якою володіє Інтернет. Мережа стала віртуальним накопиченням інформації. Будь-яка тема доступна в мережі Інтернет. Пошукові системи, як Google, надають широкі можливості користувачам мережі. Ви можете розшукати будь-який тип даних і отримати відповідь на будь-яке питання. Студенти та діти відносяться до числа найбільш активних користувачів, які займаються серфінгом в мережі Інтернет у пошуках інформації. Сьогодні Інтернет є необхідністю для якісного навчання.

3. Розваги

Розваги є найпопулярнішою перевагою мережі Інтернет. Справді, будучи засобом масової інформації, Інтернет став досить успішним інструментом в сфері розваг. Коли люди звертаються за пошуком до мережі, вони зможуть знайти що завгодно. Існує безліч ігор, які можна завантажити з мережі абсолютно безкоштовно. Індустрія онлайн-ігор відчуває величезне зростання, постійно залучаючи мільйони гравців. Музика, фільми, новини і багато чого іншого, все є в мережі.

4. Послуги

Сьогодні безліч послуг перебралися в Інтернет. Серед них: оплата житлових послуг, пошук роботи, покупка квитків, бронювання готелів та медичних засобів, надання консультацій по безлічі питань, що охоплюють всі аспекти життя та багато чого іншого.

5. Електронна торгівля

Електронна комерція – поняття, яке використовується для будь-якого виду комерційної діяльності, або ділової пропозиції, та передбачає організацію передачі товарів та інформації по всьому світу за допомогою мережі. Інтернет залучає в себе дивовижний і широкий спектр продуктів, від побутових потреб до технологій і розваг. [13]

Найпопулярнішими службами Інтернету є: веб-форуми, блоги, вікі-проекти, електронні платіжні системи, Інтернет-телебачення, Інтернет-аукціони, Інтернет-час, Інтернет-події, FTP-сервери, електронна пошта та розсилки, групи новин, Інтернет-магазини, файлообмінні мережі, Інтернет-радіо, IP-телефонія, системи обміну повідомленнями, IRC.

Основні служби доступу до віддалених файлів:

Інформаційні архіви – це аналоги комп'ютерних бібліотек, де зберігаються різноманітні програми та матеріали, що тривалий час користуються попитом та нерідко мають великий обсяг. Інформація в таких архівах зберігається у вигляді файлів, які в свою чергу в залежності від їх тематики згруповані в каталоги та підкаталоги, неначе створюючи деревоподібну структуру (аналогічно до звичної файлової структури ОС). [14] Каталог найвищого рівня називається кореневим каталогом архіву.

Основною метою поштових серверів інформаційних архівів є пересилання за запитом користувачів потрібних файлів за допомогою електронної пошти. Під час підготовки до надсилання поштовий сервер автоматично виконує (у випадку необхідності або за запитом користувача) операції кодування та сдавлення даних. Такі поштові програмні сервери можна поділити на три види:

1. файловий сервер;
2. FTPmail-сервер;
3. WAISmail-сервер.

Простий файловий сервер обслуговує лише один інформаційний архів і призначений для автоматичного прийому запитів користувачів та надсилання електронною поштою результатів виконання цих запитів у вигляді текстових файлів, графічними продуктами, програмами тощо.

На відміну від простого файлового сервера, FTPmail-сервер надає можливість користувачу отримати файли з інформаційних архівів, які працюють в режимі FTP (File Transfer Protocol). По запиту сервер автоматично складає програму взаємодії з архівом та відсилає отримані результати користувачу

електронною поштою. Увесь доступ до файлів віддалених серверів керується протоколом передачі даних File Transfer Protocol, а в разі необхідності повного обміну комп'ютер потрібно приєднати до FTP-серверу вузлового сервера, де зберігаються необхідні файли, що призначені для передачі інформації. Багато загальнодоступних FTP-серверів надають можливість під час зв'язку отримати будь-які файли, необхідні користувачу.

WAISmail-сервер (Wide Area Information Service) є довідково-пошуковим та використовується для знаходження інформаційних архівів, які містять потрібні файли.

Підсумовуючи вищесказане, можна зробити висновок, що в наш час Інтернет відіграє надважливу роль для створення інформаційного середовища глобального суспільства та слугує головною основою доступу до веб-сервісів та багатьох систем передачі інформації. Станом на грудень 2019 року у всьому світі Інтернетом активно користуються 4,3 млрд людей. Загальний рівень зріс до 57,6%. 49% жінок використовуть Інтернет для своїх потреб, а серед чоловіків відсоток ще більший – 59%. Рівень підключення зріс на 5,4% у порівнянні з попереднім роком. Найвищим він є у Європі(83,5%), а найнижчим – в Африці(28,2%). Найвищий розрив між представниками різної статі в Африці, в Азіатсько-Тихоокеанському регіоні та в арабських країнах.

1.2 Файлові менеджери та огляд існуючих рішень

Файловий менеджер – комп'ютерна програма, що надає необхідний інтерфейс користувачу для роботи з каталогами та файлами. Файловий менеджер дозволяє виконувати основні операції з файлами – створення, відкриття та переміщення, перегляд, редагування, видалення, перейменування, пошук файлів, копіювання, зміна атрибутів та властивостей та призначення прав. Файлові менеджери забезпечують більш зручний і наочний спосіб спілкування з ПК в порівнянні з операційною системою (ОС). [5]

Одна з найвідоміших перших програмних оболонок називалася Norton Commander. Її розробив американський програміст Пітер Нортон. Цей файловий

менеджер вже наочно показував на моніторі усю файлову архітектуру пристрою: диски, каталоги та файли. З такою програмою не треба було набирати складні команди MS-DOS в командному рядку. Файли можна було копіювати, переміщувати, розшуковувати, видаляти, сортувати, змінювати, запускати, користуючись лише кількома клавішами. В даний час в операційній системі Windows є свій засіб візуальної роботи з файловою системою – вбудована програма Провідник. Однак, незважаючи на це, файлові менеджери продовжують користуватися великою популярністю. Таким чином, файлові менеджери – спеціальні програми, призначені для полегшення спілкування користувача з командами операційної системи. Це програми, що запускаються під управлінням ОС і займають проміжне положення між нею та прикладними програмами.

Крім основних функцій, багато файлових менеджерів включають ряд додаткових можливостей: роботу в мережі (FTP, NFS); резервне копіювання; можливість налаштування інтерфейсу; доступ до системної інформації; можливість створення макрокоманд; архівацію; управління принтерами та іншими зовнішніми пристроями.

Виділяють основні типи файлових менеджерів:

навігаційні та просторові – інколи можлива підтримка перемикань між цими режимами;

двупанельні – зазвичай облаштовані двума рівноцінними панелями для списку файлів та дерева каталогів.

Звичайний Провідник Windows є представником навігаційних (просторових) файлових менеджерів. Це ненайкраща але зручна програма. Вона не завжди справляється з масовими операціями з файлами, наприклад, коли треба перенести або перейменувати сотню фотографій. При роботі з дуже великою кількістю файлів Провідник значно сповільнює роботу системи. Але найсуттєвіший недолік – це, все ж, відсутність другої панелі для копіювання або переміщення файлів. До інших відомих навігаційних (просторових) файлових менеджерів відносяться: Pocket Total Commander, Directory Opus Magellan,

Стандартний File Manager, Resco Explorer, Workbench - поставляється з AmigaOS, Konqueror – поставляється з KDE, Dolphin – поставляється з KDE 4, Nautilus (файловий менеджер) – поставляється з GNOME, Thunar – поставляється з Xfce, Bynarys Smart Explorer, ROX-Filer – поставляється з ROX Desktop.

У менеджерах другого типу (двохпанельних) для багатьох операцій вікно програми розділене на дві частини. [5] При цьому робота з файлами стає більш зручною та швидкою. А якщо врахувати, що більшість файлових менеджерів дозволяють управляти всіма діями з клавіатури, то швидкість роботи підвищується в кілька разів. Найбільш відомі ортодоксальні (двохпанельні) файлові менеджери: Norton Commander, Total Commander, DOS Navigator, Volkov Commander, PIE Commander, Microsoft Windows, FreeCommander, Nomad.NET, Unreal Commander, File Navigator, GNOME Commander. Розглянемо найбільш відомі файлові менеджери більш докладно.

1.2.1 Norton Commander(рис. 1.2)

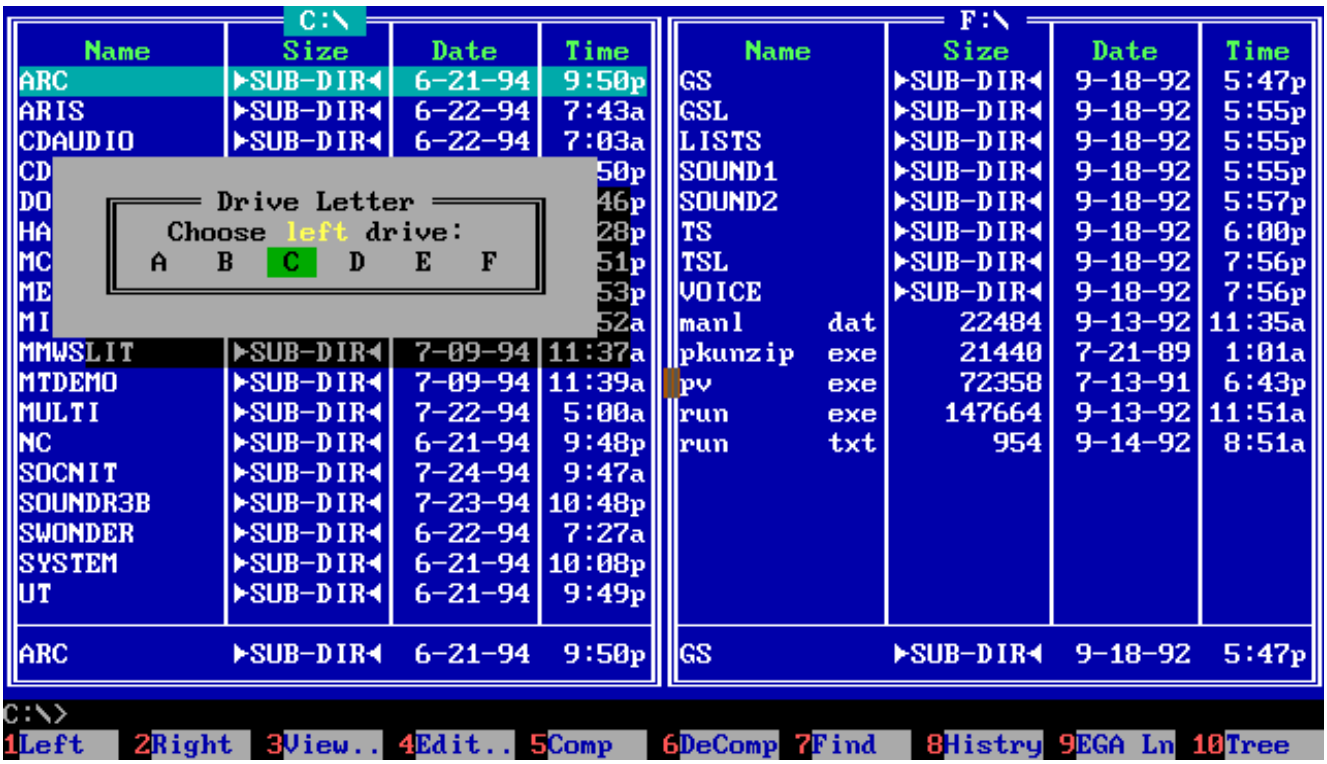


Рис. 1.2 Інтерфейс файлового менеджера Norton Commander

Norton Commander призначений для користувачів, що працюють з великою кількістю файлів та здійснюють завантаження на різні FTP-сервера, локальні

сервера, локальні диски, або свої сайти. Файловий менеджер Norton Commander має сервіс FTP-клієнт. Вміст каталогів FTP-сервера показується аналогічно до ваших власних. Можна файли копіювати, видаляти, переміщувати. Можлива і пряма передача файлів з сервера на сервер, а також продовження перерваного закачування. Norton Commander дуже корисна програма для швидкого копіювання великих обсягів даних. Для цього достатньо вибрати необхідні файли та перетягнути їх на будь-який диск або директорію в іншій панелі. Файловий менеджер Norton Commander працює з зовнішніми накопичувачами: USB і карти пам'яті. Поряд з цим ефективним способом організації файлової системи Norton Commander має вбудовані засоби відтворення медіа-файлів і можливості перегляду зображень. Робить перетворення WAV в MP3. Можна створювати образи дисків, записати їх на CD і DVD і працювати з популярними форматами архівів. Наявні багато інших корисних функцій. Таким чином, файловий менеджер Norton Commander стане прекрасним доповненням до Windows і зробить управління файлами більш швидким та зручним.

1.2.2 Програма Провідник(Explorer) Windows

Програма Провідник Windows – це спеціальний плагін, що належить до виду диспетчерів файлів та вбудований у операційну систему Windows. Програма спеціально призначена для навігації по файловій структурі Windows та її обслуговування (копіювання, перейменування, видалення і переміщення папок та файлів). Програма запускається командою Пуск/Програми/Провідник. Провідник Windows одночасно відображає як структуру вкладеності папок на комп'ютері (їх ієрархію) та показує нутро виділеної папки. Цим дуже зручно користуватись при копіюванні та переміщенні даних: досить відкрити папку, яка містить потрібний файл та перетягнути його в папку призначення. У разі, коли потрібно відкрити, скопіювати, перемістити, видалити, перейменувати або змінити іншим чином папки або файли, можна або скористатися програмою Провідник, або працювати безпосередньо на Робочому столі з папками та файлами. Основною перевагою, на відміну від звичайних вікон папок на

Робочому столі, є те, що програма Провідник надає можливість одночасної роботи як з вмістом правої панелі вікна, так і з усією архітеткурою файлової системи комп'ютера користувача, тобто з лівою панелю. Це, наприклад, дуже сильно спрощує копіювання файлів з правої панелі у каталог або на логічний диск, що в цей час відображені у лівій панелі. При цьому немає необхідності відкривати велику кількість вікон, як це довелося б робити на Робочому столі. Робота з утілітою Провідник мало чим відрізняється від роботи із звичайними вікнами Windows, за винятком того, що користувач має змогу паралельно переглядати ієрархічну структуру його файлової системи. Він може натиснути правою кнопкою миші по будь-якій папці та переглянути її вміст або, натиснувши будь-яку піктограму, перетягнути її в інше місце. Отже, Провідник – програмний засіб Windows, що призначений для гнучкого і високоефективного управління файловою системою та забезпечує зручний доступ до локальних і мережевих ресурсів.

1.2.3 Total Commander(рис. 1.3)

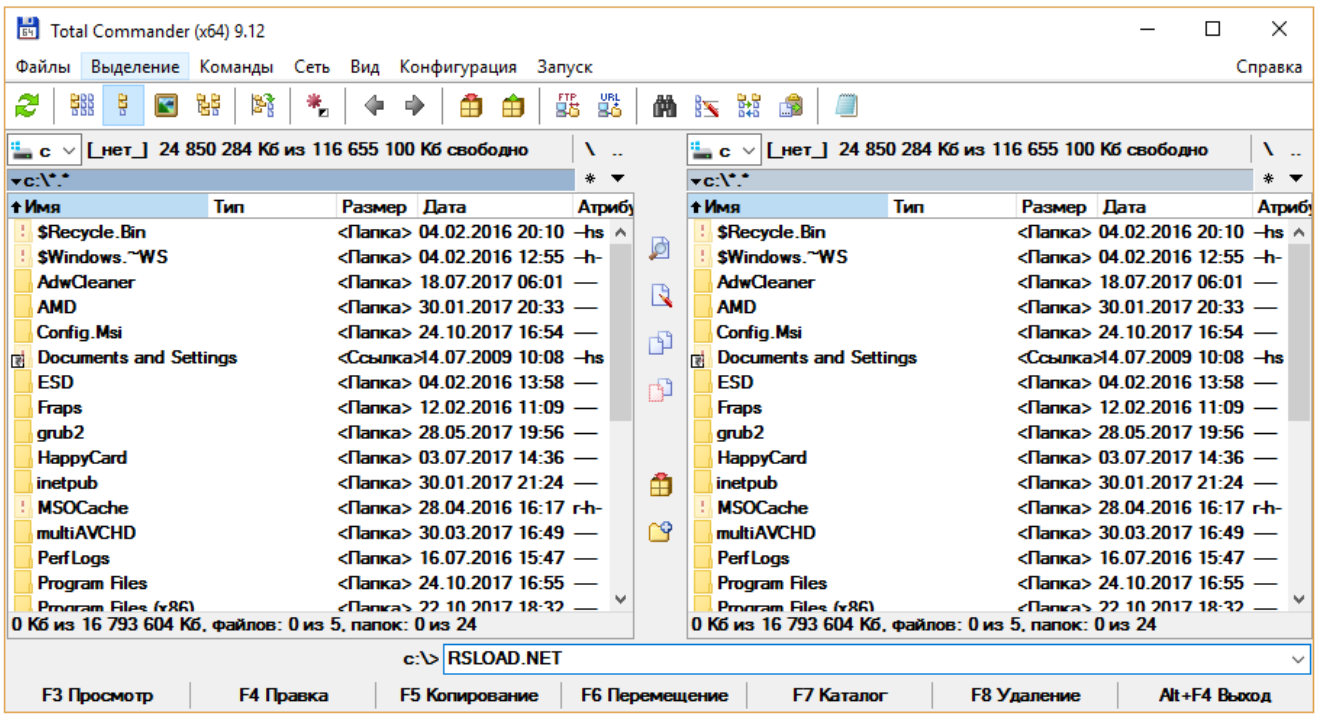


Рис. 1.3 Інтерфейс останніх версій Total Commander

Total Commander (В минулому Windows Commander) – це найпопулярніший файловий менеджер для ОС Windows, не враховуючи Провідник Windows, що призначений для більш зручної роботи з файлами і каталогами користувача. [4] Він, так само як і Провідник дозволяє виконувати всі основні операції над файлами і каталогами – копіювати, переміщувати, видаляти, перейменовувати і т. д. Однак Total Commander використовує інший, більш зручний підхід в організації інтерфейсу: у нього дві постійні панелі, які позбавляють користувача від виснажливих переходів по дереву каталогів в пошуку місця призначення для скопійованого файлу. Замість цього перед очима користувача завжди знаходяться каталог-джерело файлів і каталог-приймач файлів на той випадок, якщо користувачеві необхідно буде ще раз скопіювати файли з каталогу-приймача.

Основні можливості Total Commander:

- двопанельний багатомовний (включаючи українську) графічний інтерфейс користувача;
- панель інструментів та головне меню можуть будь-як налаштуватися самим користувачем;
- усі клавіатурні скорочення, максимально відповідають оригінальному Norton Commander з можливістю будь-як переназначити їх, водночас має місце підтримка загальноприйнятних скорочень, які наразі існують в ОС Windows;
- підтримка зручної функції Drag&Drop;
- вкладковий інтерфейс;
- можливість роботи з папками та файлами (створення, видалення, відкриття/програвання/запуск/перегляд, пошук файлів, редагування, переміщення, перейменування, копіювання, зміна атрибутів та властивостей, призначення прав);
- одночасне масове перейменування файлів та синхронізація каталогів (але, на відміну від rsync, відсутні звичайні та розширені атрибути);

- порівнювання файлів з можливістю порівняння та редагування файлів з різними кодовими сторінками;
- повна підтримка як послідовної черги, так і фонового паралельного виконання необхідних операцій над файлами (видалення, копіювання, перенесення, робота з архівами чи FTP);
- журналювання усіх виконаних файлових операцій;
- можливість роботи з довгими шляхами для NTFS;
- підтримка архівів TAR, ZIP, TGZ, GZ (розпакування і запакування) та RAR, 7-Zip, ACE, ARJ, LZH, UC2, (розпаковування) та робота з ними як з підкаталогами;
- підсвічування та виділення файлів на панелі за розширенням, іменем, часом та датою створення файлу, багатьма іншими умовами;
- наявний командний рядок для звичайного запуску програм з вказаними параметрами;
- можливість розширеного пошуку файлів включає пошук тексту всередині та у назві файлу, пошук файлів за розміром, датою, шаблонами, властивостями тощо; підтримується пошук файлів у архівах;
- можна використовувати регулярні вирази під час пошуку файлів;
- вбудований переглядач файлів, у тому числі графічних;
- наявний FTP-клієнт, який надає змогу завантажувати/відвантажувати файли одразу в декілька потоків та шукати файли на FTP-серверах;
- XXE/MIME/UUE кодування/декодування та склеювання/розрізання довгих файлів;
- перевірка та підрахунок контрольних сум (MD5, SHA1, CRC32);
- змога тимчасово підвищити рівень привілеїв;
- розширення функціональності програми з допомогою скриптів (PowerPro, AutoHotKey) та додаткових модулів – плагінів. [12]

1.2.4 Файловий менеджер Frigate(рис. 1.4)

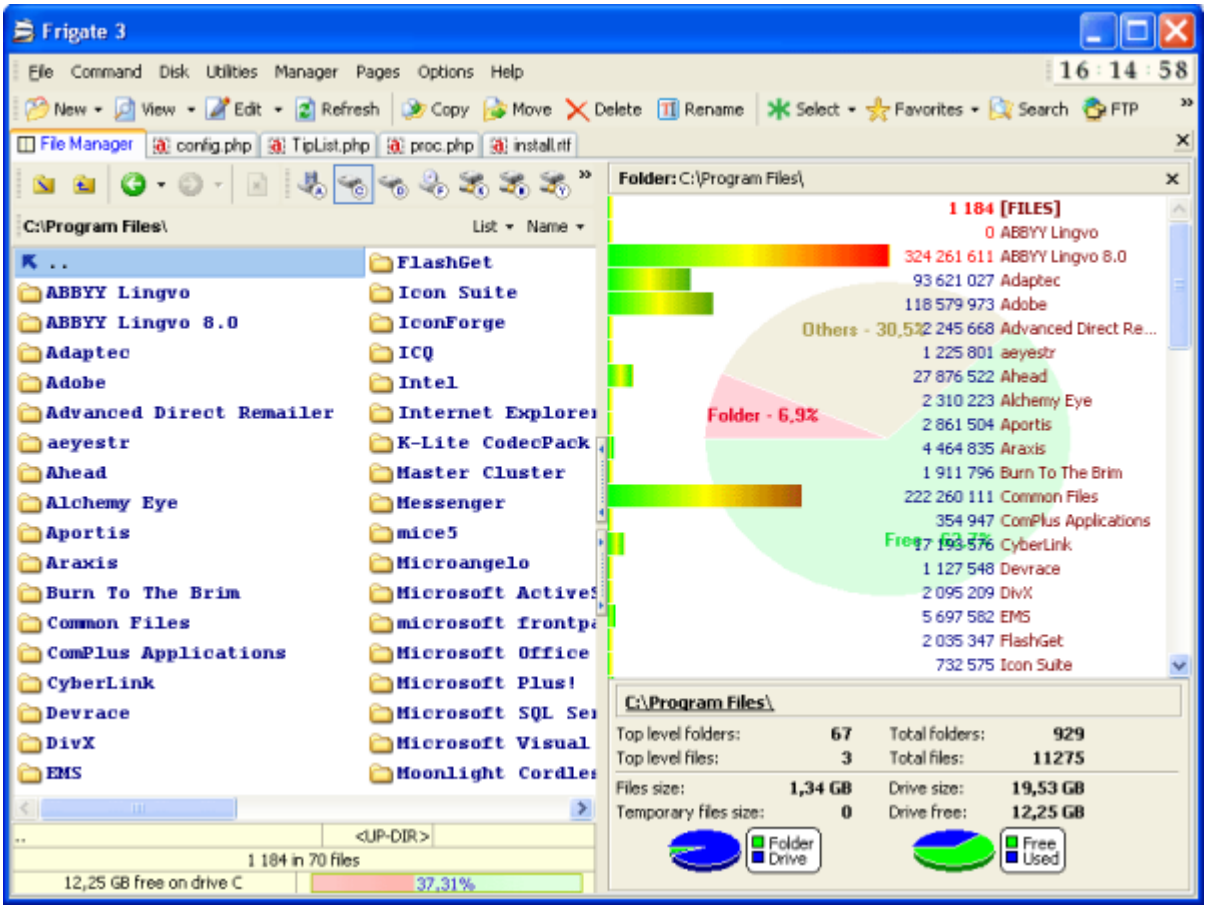


Рис. 1.4 Інтерфейс Frigate

Frigate – це дуже потужний файловий менеджер, що дозволяє швидко, інтуїтивно-зрозуміло і, найголовніше, з комфортом працювати зі своїм комп'ютером. На сьогоднішній день Frigate є самим широкофункціональним та найпотужнішим засобом роботи з файлами. При першому запуску Frigate в першу чергу привертає увагу своїм приємним та сучасним графічним оформленням. Що стосується функціональності, то Frigate багато в чому схожий на інші файлові менеджери, проте є і відмінності. Основна з них – підтримка багатозадачності, яка тут реалізована найбільш зручно. Наприклад, можна одночасно копіювати файли, закачувати їх на сервер, і в цей же час працювати з директоріями. [14]

Деякі характеристики Frigate:

- Багатозадачність при виконанні файлових операцій;
- Підтримка усіх відомих архівів;
- Зручна навігація – "швидкі" папки, наявна історія переміщення по папках;

- Наявність корисних утиліт (калькулятор, системна інформація, телефонна книга, зберігач паролю, відправка пошти та вікно перегляду буфера обміну);
- Вбудований перегляд (F3) для RTF-тексту і файлів DBF (працює в FTP і архівах);
- Вбудований редактор (F4) для тексту з підсвічуваним синтаксисом і RTF-текстовими файлами;
- Відновлення файлів після обриву копіювання;
- Перегляд і редагування MP3-тегів;
- Вдосконалений пошук;
- Файловий фільтр, завдяки якому можна швидко відібрати потрібний тип файлів;
- Робота з картинками, підтримуються всі популярні графічні формати (JPEG, PNG, GIF і т. д.).

Файловий менеджер Frigate є хорошим прикладом широкої функціональності та гнучкості, яка забезпечується його багатими можливостями в плані налаштування і конфігурації. Нова архітектура, заснована на технології ActiveX, дозволила зробити Frigate по-справжньому модульним, завдяки чому практично будь-яка частина системи, за винятком деякої частини ядра, працює незалежно і може бути при необхідності відключена. Подібний підхід до архітектури програми позитивно позначився на її продуктивності та надійності. Frigate має сучасний інтерфейс та легко піддається налаштуванню. Програма підтримує колірні схеми, що дозволяють практично повністю змінити її зовнішній вигляд для досягнення максимальної комфортності та ефективності роботи. Даний файловий менеджер має багатомовний інтерфейс. Програма підтримує концепцію закладок і дозволяє відкрити декілька файлових панелей, вікон редагування та перегляду файлів і перемикатися між ними за допомогою миші та гарячих клавіш. Файлові операції здійснюються у фоновому режимі. Є спеціальна черга копіювання, що дозволяє організовувати відповідні операції

так, щоб вони починалися тільки після завершення інших. Основна перевага і гідність Frigate полягає в тісній інтеграції всіх його модулів і додаткових утиліт, за рахунок чого вдається домогтися високої ефективності та зручності роботи. Користувач, не перемикаючись з клавіатури на мишу, може виконати практично будь-яку дію, оскільки фокус весь час залишається в програмі, що сприяє підвищенню продуктивності праці. Ідеологія плагінів, що розширюють можливості Frigate, полягає в реалізації деякої мінімальної розумної функціональності, необхідної більшості користувачів. При цьому лише замість декількох з них можливо знайти та використати більш потужні зовнішні програми. По суті, Frigate є не просто файловим менеджером, а цілим офісом, що володіє масою можливостей і здатний замінити десятки інших програм. До складу професійної комплектації входить більше 25 модулів та різних плагінів, а також ряд інших операцій, які підтримуються ядром програми.

Також Frigate має значну кількість вбудованих корисних утиліт. [5]

Менеджер автозапуску, що дозволяє управляти автозапуском програм. Може стати в нагоді як для того, щоб переконатися, що при запуску Windows не завантажується нічого зайвого, так і при необхідності вручну додати потрібну програму.

Швидкий блокнот, по суті, являє собою простий текстовий редактор з тією лише різницею, що запускається він за допомогою спеціальної комбінації клавіш. Може використовуватися для зберігання тимчасових заміток або в якості тимчасового редактора.

Вбудований калькулятор виручить в тих ситуаціях, коли потрібно щось швидко порахувати. Для зручності розряди в числах можна розділяти пробілами, а як роздільник між цілою і дробовою частиною в десяткових дробах підтримуються і кома, і крапка.

Структурний блокнот – це інструмент для зберігання різних записів або упорядкування завдань. Кожну замітку можна відредагувати в редакторі, що підтримує зображення, форматування тексту і кольорове виділення.

Ще одним корисним інструментом є конвертор картинок, за допомогою якого можна не тільки змінити формат зображення, але і його розмір. Утиліта підтримує роботу з вісьмома найбільш поширеними графічними форматами.

Frigate містить вбудовані аудіо- і відеопрогравачі. Звичайно, вони не такі потужні, як деякі програми, але цілком підійдуть для того, щоб знайти потрібний мультимедійний продукт, не викликаючи для кожного кандидата зовнішню програму.

До того ж з офіційного сайту можна скачати Frigate 3 Extension Pack, що містить дев'ять додаткових програм, які ще більше розширюють функціональні можливості цього файлового менеджера. [14] Після його установки в меню з'явиться пункт Tools, що забезпечить зручний перегляд та запуск даних інструментів:

За допомогою ImageBox можна створити слайд-шоу або презентацію з зображень і музичних файлів.

Утиліта Image2Ico призначена для створення і редагування іконок.

Програма CDuke дозволяє записувати CD- та DVD-диски і має оптимізований інтерфейс для різних типів дисків.

Програма EasyDVDCopy дозволяє копіювати DVD-фільми.

Утиліта SecureWallet призначена для зберігання паролів та іншої персональної інформації, такої як серійні номери до програм і дані кредитних карт.

Програма VisualRenamer дозволяє перейменувати цілі групи файлів та папок.

Модуль PrintMaestro незамінний для тих, кому треба швидко надрукувати або перевести структуру директорій та їх зміст в PDF- або HTML-файли.

Утиліта FolderShine дозволяє легко привласнити іконки найбільш важливим папок. У комплект входить більше 200 красивих, професійно виконаних іконок.

Програма AudioConverter призначена для конвертування музичних файлів.

Висновки до розділу 1

Сучасні інтернет-сервіси, що дозволяють перенести обчислювальні ресурси та дані на віддалені сервери, останними роками стали одним з основних напрямів розвитку IT-технологій.

Розповсюдження мереж з високою потужністю, значна вартість комп'ютерів та інших пристроїв зберігання даних, широке впровадження віртуалізації та сервіс-орієнтованої архітектури призвели до величезного зростання актуальності інтернет-технологій.

Недоліки таких рішень в основному зводяться до проблеми довіри клієнта постачальнику сервісу, адже від нього повністю залежить безперебійна робота, та збереження важливих даних користувача. Крім того, на даний момент усі подібні обчислення висувають зависокі вимоги до якості каналів зв'язку, адже вони повинні гарантувати повсюдний та якісний доступ до Інтернету.

Щодо файлових менеджерів, то незважаючи на значне покращення їх роботи та розвиток функціональних можливостей, вони не мають значної популярності через відсутність значної необхідності їх використання. Але можливо завдяки інтеграції файлових менеджерів в сучасні інтернет-технології нам вдасться повернути актуальність подібних продуктів.

РОЗДІЛ 2

АНАЛІЗ ЗАСОБІВ РЕАЛІЗАЦІЇ

2.1. Вибір мови програмування

Мовою програмування було обрано Python(рис. 2.1) і в цьому підрозділі коротко викладені основні її переваги.

Рис. 2.1 Офіційний логотип Python

Python – високорівнева мова програмування широкого призначення, орієнтована на підвищення продуктивності розробника та полегшення читання коду. Основні архітектурні риси – динамічна типізація, механізм обробки виключень, автоматичне керування пам’яттю, підтримка багатопоточних обчислень та повна інтроспекція. Динамічні семантика та зв’язування, а також структури даних високого рівня роблять цю мову привабливою для пришвидшеної розробки програм. [6]

Python є мовою, що підтримує велику кількість парадигм програмування, зокрема: об’єктно-орієнтована, процедурна, імперативна, функціональна та аспектно-орієнтована. Як і більшість інтерпретованих мов, Python є кросплатформеною і не вимагає додаткових зусиль для перенесення з однієї платформи на іншу. Інтерпретатор написаний на C і вихідний код доступний для будь-яких маніпуляцій. У випадку необхідності можна вставити його в свою програму та використовувати як вбудовану оболонку. Або ж, написавши на C свої доповнення, отримати “розширений” інтерпретатор з новими можливостями.

Python також зручний у якості мови розширення для прикладних програм, яким необхідне подальше налагодження. Мова підтримує модулі та пакети

модулів, що значно підвищує повторне використання коду і його модульність, що є корисним при розробці великих додатків. Інтерпретатор Python та стандартна бібліотека доступні в скомпільованій та у вихідній формах на всіх основних платформах.

Багата стандартна бібліотека відіграє окрему роль у привабливості Python. Вона включає широкий вибір засобів, які будуть корисними для абсолютно різних галузей праці, що робить її зручною, при використанні в складних багатоцільових проектах. [7] Тут наявна велика кількість засобів для роботи з багатьма мережевими форматами та протоколами Інтернету, наприклад, модулі для написання HTTP-клієнтів та серверів, можливість створення та розбору поштових повідомлень, плагіни для комфортної роботи з XML та багато іншого. Набір спеціальних модулів для роботи з ОС дозволяє легко створювати кросплатформні застосунки. Існують модулі для роботи з текстовими кодуваннями, мультимедійними форматами, юніт-тестуванням, криптографічними протоколами, регулярними виразами, архівами, серіалізацією даних та ін.

Окрім вбудованої існує багато додаткових бібліотек, які надають інтерфейс до всіх системних викликів навіть на різних платформах; наприклад, на платформі Win32 підтримуються усі виклики Win32 API та COM не менше, ніж у Visual Basic або Delphi. В Інтернеті наявна велика кількість прикладних бібліотек для Python разом з детальними інструкціями для використання у різноманітних галузях: чисельні методи, обробка зображень, веб-розробка, бази даних, програми операційної системи, обробка тексту тощо.

Для цієї мови програмування прийнята специфікація програмного інтерфейсу для баз даних DB-API 2, також розроблено відповідні цій специфікації пакети для надання доступу до різних СУБД: Oracle, Sybase, sqlite, Firebird (Interbase), MySQL, Informix, Microsoft SQL Server, PostgreSQL тощо. За допомогою ADO (ADOdb) можливий доступ до БД на платформі Microsoft Windows. Більш того для Python написано багато ORM (Dejavu, SQLAlchemy,

SQLObject), наявні програмні каркаси для зручної розробки веб-застосунків (Django, Pylons).

Дуже популярною є бібліотека NumPy, що призначена для роботи з багатовимірними масивами та дозволяє досягти продуктивності у наукових розрахунках, порівнянної зі спеціалізованими платними пакетами. SciPy активно використовує NumPy для надання ще більш широкого доступу до значної кількості математичних алгоритмів.

Бібліотека WSGI (Python Web Server Gateway Interface) є комфортним інтерфейсом шлюзу з веб-сервером.

Python дозволяє використовувати код скомпільований на мові C та C++, це значно розширює використання мови, крім того дозволяє переносити важкі обрахунки на більш швидку мову C та надає зручний та простий програмний інтерфейс C API для створення модулів на мовах C++ та C.

Інструмент, що має назву SWIG, надає можливість майже автоматично отримати прив’язки для використання C/C++ бібліотек у коді мови Python. Інший інструмент стандартної бібліотеки ctypes дає змогу програмам Python безпосередньо викликати процедури з динамічних бібліотек/DLL, написаних на C. Також існують модулі pyinline та weave, що дозволяють вбудовувати код написаний на C/C++ прямо у сирцеві файли Python, створюючи так зване розширення «на льоту». Наразі офіційно рекомендовано Cython разом із застосуванням NumPy для підключення математичних функцій. Інша перевага полягає у легкому вбудовуванні Python у програми на C/C++, Java та Ocaml. Взаємодія Python-застосунків з іншими можлива також завдяки COM, XML–RPC, CORBA, SOAP.

Python та переважна більшість бібліотек до нього поставляються у вихідних кодах та є безкоштовними. [6] Більш того, на відміну від багатьох відкритих систем, ліцензія не накладає ніяких зобов'язань та не обмежує використання Python для комерційних розробок крім зазначення авторських прав.

Також надається можливість скористатись бібліотекою tkinter на основі Tcl/Tk для створення інтерфейсу в кросплатформених програмах.

Для наукових дослідів значного поширення набула matplotlib – це бібліотека з інтерфейсом, аналогічним до популярного MATLAB Plot Tool.

Розширення wxPython, засноване на бібліотеці wxWidgets, PySide та PyQt для Qt, PyGTK для GTK+ дозволяють використовувати ці GUI бібліотеки. Також багато з них надають широкі можливості для роботи з графікою, мережами та базами даних, користуючись всіма можливостями бібліотеки, на основі якої вони створені.

Для створення програм та ігор, що потребують незвичного інтерфейсу, можна скористатись бібліотекою Pygame, яка також надає значну кількість засобів роботи з мультимедіа. З її допомогою можна відтворювати відео, керувати звуком і зображеннями, тощо. Апаратне прискорення графіки OpenGL, що запропоноване Pygame має більш високорівневий інтерфейс, якщо порівнювати з популярним PyOpenGL, який копіює семантику C-бібліотеки для OpenGL. Існує також PyOgr, забезпечуючи прив'язку до OGRE, що є високорівневою об'єктно-орієнтованою бібліотекою 3D-графіки. Крім того, прив'язку до середовища 3D-моделювання та симуляції OpenCascade забезпечує інша бібліотека pythonOCC. Вдало можна використовувати Python Imaging Library для роботи з растровою графікою.

Ще одним значним плюсом мови є те, що код є інтуїтивно зрозумілим і читабельним. Єдиним недоліком Python є порівняно невисока швидкість виконання Python-програми, що обумовлено її інтерпретованістю. Проте переваги значно більші у програмах не дуже критичних щодо швидкості виконання. Також, цей недолік компенсується зменшенням часу, необхідного для розробки програми. У середньому, Python-програма в 2-4 рази компактніша, ніж аналоги на Java та C++. Збереження байт-коду (файли .pyo та .pyc) надають можливість інтерпретатору не витрачати зайвий час на повторну перекомпіляцію коду декількох модулів при кожному запуску програми, на відміну від,

наприклад, мови Perl. На додачу, існують проекти реалізацій мови Python, які застосовують високопродуктивні віртуальні машин як компілятори заднього плану. Прикладами подібних реалізацій є PyPy, що базується на LLVM; більш ранньою розробкою є Parrot.

Python, як вже було згадано, підтримує динамічну типізацію. Це означає, що тип змінної формується автоматично під час виконання. З базових типів окремо слід зазначити підтримку комплексних чисел та цілих чисел довільної довжини. В цій мові також наявні засоби для роботи з рядками, зокрема і кодованими в юнікоді. З колекцій Python підтримує списки (масиви), словники (асоціативні масиви), кортежі (*tuples*) та з версії 2.4, множини.

Дизайн Python сформований довколо об’єктно-орієнтованої моделі програмування. Реалізація ООП гарно продумана, потужна та елегантна, але разом з тим, доволі специфічна та може бути незвичною для розробників звикших до інших об'єктно-орієнтованих мов.

Можливості та особливості Python:

- класи одночасно є об’єктами з усіма відповідними наслідками та можливостями;
- присутнє успадкування, зокрема множинне;
- виконується парадигма поліморфізму;
- виконується парадигма інкапсуляції (можливі два рівні – приховані та загальнодоступні поля і методи);
- спеціальні методи, що надають можливість керувати життєвим циклом об’єкта: розподільники пам’яті, конструктори, деструктори;
- можливе перевантаження операторів (усіх, крім `.`, `=`, `is` та символьних логічних);
- керування доступом до полів (частковий доступ, емуляція методів та полів);
- методи керування найпоширенішими операціями (ітерація по об’єкту, len(), глибоке копіювання, серіалізація, істинносне значення та ін.);

- можливість метапрограмування (керування створенням класів, тригери на створення класів);
- наявність повної інтроспекції (можна отримати необхідну інформацію про внутрішню будову будь-якого об’єкта);
- класи, вкладені у інші класи та функції, класові поля, статичні та класові методи;
- підтримка множинного успадкування системою класів, адже будь-який тип, включаючи базові, належить до системи класів і, за необхідності, є можливим успадкування навіть від базових типів.

На додачу, Python підтримує парадигму функціонального програмування, а саме:

- функція також є об’єктом;
- можливість створення функцій вищих порядків;
- можливість рекурсії;
- добре розвинута обробка списків (ітератори, операції над послідовностями, спискові вирази);
- наявний аналог замикань (closures);
- можливе часткове застосування функції.

Станом на травень 2020 року Python займає третє місце в рейтингу TIOBE з показником 9,12%(рис. 2.2).

TIOBE Index for May 2020

May 2020	May 2019	Change	Programming Language	Ratings	Change
1	2	▲	C	17.07%	+2.82%
2	1	▼	Java	16.28%	+0.28%
3	4	▲	Python	9.12%	+1.29%
4	3	▼	C++	6.13%	-1.97%
5	6	▲	C#	4.29%	+0.30%
6	5	▼	Visual Basic	4.18%	-1.01%

Рис. 2.2 Рейтинг популярності мов програмування

2.2 Вибір фреймворку для веб-інтерфейсу

Для виконання поставленого завдання програмний продукт повинен мати клієнт-серверну архітектуру. Функції вузлів такої системи та зв’язки між ними продемонстровано в додатку А. Відповідно нам потрібно використати якісний та зручний засіб створення веб-сервісів. Було обрано рішення, що фреймворк Django(рис. 2.3) найкраще відповідає необхідним вимогам.

Рис. 2.3 Офіційний логотип Django

Django – високорівневий відкритий Python-фреймворк для розробки веб-систем. Тобто, це програмне забезпечення, яке підтримує розробку динамічних веб-сайтів, застосунків і служб. Він надає набір інструментів і функціональних можливостей, які вирішують багато загальних проблем, пов’язаних із розробкою веб-сайтів: безпека, доступ до БД, сесії, обробка шаблонів, маршрутизація URL-адрес, інтернаціоналізація, локалізація тощо. [10] Сайт на Django зазвичай побудований з однієї чи кількох частин, які рекомендовано робити модульними.

Django з’явився в 2005 році, і поступово став одним з кращих фреймворків, який допомагав і допомагає тисячам розробників виконувати ту чи іншу роботу протягом декількох хвилин. Спочатку Django був фреймворком для мови Python, з відмінним функціоналом, Django помітно спростив ряд складнощів в розробці веб-додатків, надавши цій роботі більш спрощений підхід.

Архітектура Django дещо схожа на звичну «Модель-Вид-Контролер» (MVC). Однак те, що позначається «контролером» в звичній моделі MVC, в Django перетворюється на «вид» (англ. *view*), а те, що повинно було бути «видом», називається «шаблон» (англ. *template*). Керуючись цим, розробники Django для позначення архітектури використовують назву MTV («Модель-Шаблон-Вид»).

Спочатку Django розроблявся як засіб для роботи новинних ресурсів, що значною мірою позначилося на його структурі: він надає цілий ряд зручних засобів, які сильно прискорюють розробку веб-сервісів інформаційного характеру. [11] Програмісту вже не потрібно створювати сторінки для адміністративної частини сайту і контролери, в Django вже є вбудовані модулі, які можна легко підключити до будь-якого сайту, розробленого на Django.

Наприклад адміністративний модуль надає змогу швидко змінювати, створювати та вилучати будь-які об’єкти з сайту, протоколюючі всі ці дії, а також надаючи інтерфейс для управління користувачами і групами (з повноцінним призначенням прав).

Django підтримує парадигму ООП. [9] Об’єкти баз даних в термінології цього фреймворку називаються «моделями». Django пропонує розробникові розвинений програмний прикладний інтерфейс для високорівневого доступу до даних. Зазвичай потреби писати SQL-запити немає, хоча ніхто не забороняє це робити.

У дистрибутиві також наявні програми для перенаправлення URL, синдикації RSS та Atom, системи коментарів, «статичних сторінок» (якими можна керувати без необхідності писати контролери та відображення) та інше.

Деякі можливості фреймворку Django:

- ORM, можливість API доступу до БД з підтримкою транзакцій;
- на основі регулярних виразів наявний диспетчер URL та можливий їх парсинг;
- Гарно розширювана система шаблонів з наслідуванням і тегами.

- система кешування;
- вбудований модуль з інтерфейсом для адміністратора з наявними перекладами на велику кількість мов;
- інтернаціоналізація;
- шаблони функцій контролерів – так звані «generic views»;
- автоматичне створення CRUD-інтерфейсу ('адмін-панель');
- існує окрема мова для опису шаблонів, яка є дуже простою і «дружньою» навіть для непрограмістів і в якій присутні оператори умови, циклу та форматування;
- згадана вище мова шаблонів виконує лише функцію відображення даних, отже, неможливо випадково змінити її операторами дані в БД;
- відповідні інструменти дають змогу кешувати шаблони та їх частини, SQL-вибірки і просто окремі змінні;
- переклад проекту на Django не є проблемою. Інтернаціоналізація користується концепцією «лінивого» перекладу. Тобто, якщо певний рядок тексту не буде мати перекладу, то замість цього використовується базовий текст і не будуть висвітлені повідомлення про помилки. Проте немає заборони користатись функціями, що контролюють наявність перекладу рядкових даних. Також для перекладу тексту прямо усередині коду застосовується функція ugettext;
- можливість аутентифікації та авторизації, підключення відповідних зовнішніх модулів: OpenID, LDAP та ін;
- включена система фільтрів (так званий «middleware») для зручної побудови додаткових обробників запитів, зокрема, наявні фільтри для підтримки анонімних сесій, стиснення, кешування та нормалізації URL;
- присутня утіліта для роботи з формами (побудова форм за існуючою моделлю БД, наслідування, тощо);

- вже вбудована автоматична документація по моделям даних та тегам шаблонів, до якої можна отримати доступ через адміністративний застосунок. [10]

Різні компоненти Django слабко пов’язані між собою, тому можливо будь-яку частину замінити на схожу. Зокрема, замість вбудованих у фреймворк шаблонів можна скористатись Jinja або Mako.

Django спочатку був спроектований для роботи під керуванням веб-сервера Apache з модулем mod python і використовуванням PostgreSQL у якості БД. На даний момент Django також підтримує mod wsgi, FastCGI та SCGI. Веб-сервером може бути CherryPy, lighttpd чи nginx. Крім вже згаданого PostgreSQL підтримується значна кількість інших систем БД – Oracle, SQLite та MySQL.

У складі цього фреймворку також присутній власний веб-сервер для налагоджування та розробки. Цей сервер моніторить зміни у сирцевому коді та автоматично перезапускається, що зручно для розробки проектів.

Те, що Django знаходиться у вільному доступі, дає можливість помітно спростити процес веб-розробки, так як розробник може сфокусуватися на процесі дизайну і функціоналу програми. Таким чином, Django – це ідеальний інструмент для стартапів, коли веб-дизайн повинен відображати концепцію і цілі компанії. Коли в вас виникає певна думка, трансформувати її на мові програмування і створити їй реальну форму за допомогою Django займе всього кілька хвилин.

Мінуси завжди йдуть за руку з плюсами. Давайте розглянемо, чому Django не можна назвати бездоганним:

• усе базується на ORM Django;

• необхідність використання шаблону маршрутизації із зазначенням URL;

• Django доволі монолітний;

• спільне рзгорнення компонентів;

• потрібно вміти володіти усією системою для якісної роботи.

Висновки до розділу 2

Для реалізації завдання бакалаврської роботи нам потрібно якісно побудувати мережеву архітектуру типу ‘‘клієнт-сервер’’. Для цього серед великої кількості засобів було обрано мову програмування Python та її фреймворк Django.

Python – стабільна та поширена мова, що використовується в багатьох великих компаніях. Вона застосовується в багатьох проектах та у різних якостях: як основна мова програмування, так і для створення інтеграцій і розширень додатків. На Python вже реалізовано значну кількість різноманітних проектів, також наразі він ще активніше використовується для створення прототипів майбутніх програм.

Django – один з найпопулярніших та найсучасніших фреймворків, що продовжує постійно розвиватись та доповнюватись. В цьому засобі для побудови веб-додатків наявне все необхідне для створення багатофункціональних сайтів. Django використано в таких великих та відомих інтернет-платформах, як YouTube, Instagram, Pinterest, Google та ін.

Переваги Python та Django очевидні, тож для виконання поставленої задачі вибір одразу ж зупинився саме на них.

РОЗДІЛ 3

ПРОЕКТУВАННЯ ПРОГРАМНОГО ПРОДУКТУ

3.1. Архітектура програмного продукту

На рис. 3.1 зображено частину проекту, яку автоматично створює Django при використанні команди `django-admin startproject project_name`.

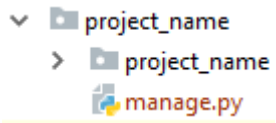


Рис. 3.1 Щойно згенерований проект Django

manage.py – інтерфейс для використання утиліти командного рядка django-admin. Він використовується для запуску команд управління, пов'язаних з нашим проектом. Зокрема його використовують для запуску сервера розробки, запуску тестів, створення міграцій та багато іншого.

project_name – набір конфігурацій(рис. 3.2).

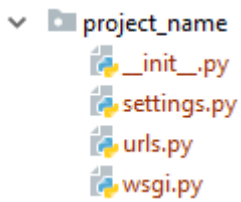


Рис. 3.2 Склад директорії project_name

__init__.py – файл повідомляє Python, що папка є пакетом Python.

settings.py – файл містить всі налаштування проекту.

urls.py – файл відповідає за відображення маршрутів і шляхів у нашому проекті. Наприклад, якщо ми хочемо показати щось в URL-адресі /about/, ми повинні прописати його тут.

wsgi.py – файл являє собою простий інтерфейс шлюзу, який використовується для розгортання.

Як було вже згадано важливою особливістю проектів Django є їх модульність. Ми можемо створювати або підключати раніше створені веб-додатки в будь-якій кількості.

Генерація нових застосунків виконується за допомогою команди `django-admin startapp app_name`(рис. 3.3).

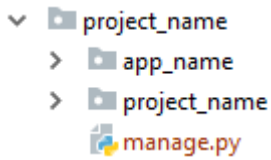


Рис. 3.3 Склад папки після створення веб-додатку app_name

На рис. 3.4 показано з яких основних частин складаються усі застосунки Django.

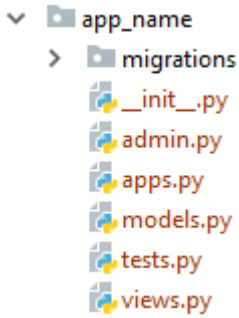


Рис. 3.4 Склад директорії app_name

Розглянемо більш детально призначення кожної з частин.

Папка *migrations* – тут Django зберігає деякі файли для відстеження змін, створених у файлі models.py, щоб підтримувати синхронізацію бази даних та моделей.

admin.py – файл конфігурації для вбудованого застосунку Django під назвою django-admin.

apps.py – файл конфігурації самого веб-додатку.

models.py – тут ми визначаємо об’єкти нашого веб-додатку. Моделі в основному являють собою макет бази даних вашого застосунку. Django автоматично переводить моделі в таблиці БД.

tests.py – файл використовується для написання модульних тестів.

views.py – файл, в якому ми обробляємо цикл запитів/відповідей.

Також в результаті розробки продукту в нас як і в більшості інших Django-проектів додадуться наступні частини програми:

Папка *templates* – в таких папках ми будемо зберігати HTML-шаблони, тобто вигляд різних сторінок та компонентів нашого сайту.

db.sqlite3 – база даних, що зберігає ряд служебних таблиць. Створені нами моделі у файлі models.py після виконання міграцій набувають вигляд таблиць та зберігаються саме тут.

backend.py – в цей файл ми окремо винесемо системну частину роботи нашого файлового менеджера.

Таким чином, після завершення архітектура продукту повинна мати приблизно наступний вигляд(рис. 3.5).

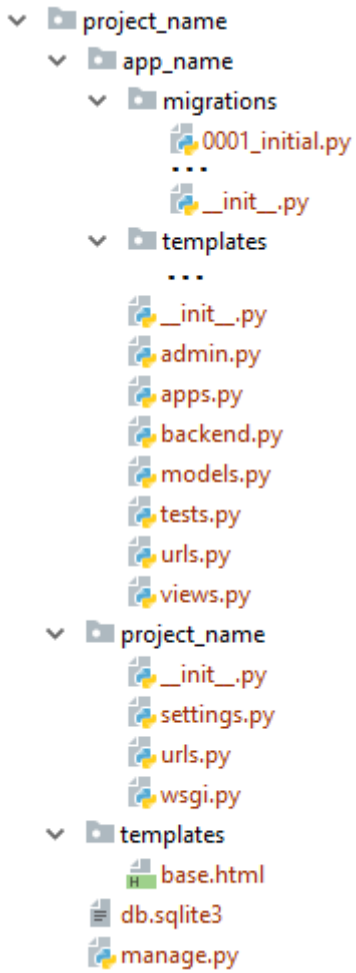


Рис. 3.5 Приблизний вигляд архітектури проекту

Більш наглядно схему взаємодії ключових модулів програми можна переглянути у додатку В.

3.2. Розбір ключових елементів програми

urls.py

Як було вже згадано файл відповідає за відображення маршрутів і шляхів у проекті. Отже саме тут ми повинні передбачити всі можливі дії користувача на сайті та передати управління та дані відповідному класу з файлу views.py, який виконає потрібне завдання. Для кращого розуміння розберемо невелику частину програми як приклад(рис. 3.6).

```
from django.urls import path
from .views import *

urlpatterns = [
    path('', main_page, name='main_page'),
    path('delete/', delete_thing.as_view(), name='delete_thing')
]
```

Рис. 3.6 Частина файлу urls.py

Django-функція *path(шлях, request-функція або view-клас, аргументи):*

`''` – коренева URL-адреса;

`main_page` – request-функція з файлу views.py, що формує головну сторінку сайту;

`name='main_page'` – ім'я для ідентифікації даної URL;

`'delete/'` – шлях, що відкривається при використанні кнопки "Видалити";

`delete_thing.as_view()` – клас з файлу views.py, що виконує обробку запиту на видалення та формує відповідь користувачу;

`name='delete_thing'` – ім'я для ідентифікації даної URL.

views.py

З попереднього прикладу стає зрозумілим, що в файлі views.py пишуться класи-представлення та функції-запроси. Розглянемо основні та найчастіше вживані компоненти програми в цьому файлі(рис. 3.7).

```
class view_class(View):
    def post(self, request):
        allparameters = dict(request.POST)
        path = allparameters.get('path')
        return render(request, 'return_page.html', {'parameter': path})
```

Рис. 3.7 Ключові частини коду з файла views.py

Розберемо окремо кожну з них:

`class view_class(View)` – наслідування базового класу View дозволяє нашому класу приймати запрос, формувати та повертати відповідь.

`def post(self, request)` – метод класу-представлення, що може бути викликаним для обробки запросу.

`request.POST, request.GET, request.FILES` – об'єкти з інтерфейсом схожим на словники(QueryDict), які зберігають передані формами параметри та дані.

`return render(request, template_name, dictionary_of_parameters)` – поєднує вказаний шаблон та необхідні дані формуючи об'єкт HttpResponse, який і відобразиться користувачу.

models.py

Для коректного та організованого відображення даних на сайті, нам необхідно створити модель, яка буде показувати необхідну інформацію про файли та папки у звичному для файлових менеджерів вигляді. Розберемо ключові елементи програмного коду на прикладі створеної для наших цілей моделі (рис. 3.8):

```
from django.db import models
class Data(models.Model):
    slug = models.SlugField(max_length=150, blank=True, unique=True)
    name = models.TextField(blank=True)
    size = models.TextField(blank=True)
```

Рис. 3.8 Приклад створення простої моделі

`models.Model` – наслідування цього класу показує Django, що клас Data є моделлю, яку необхідно зберегти в базі даних.

`models.SlugField(), models.TextField()` – ці та багато інших полів разом зі своїми параметрами призначені для зберігання усієї потрібної інформації про конкретну модель, зокрема її унікальний ключ, текст, дата створення та багато іншого.

Папка *migrations*

Після створення моделі нам необхідно зберегти її у нашій базі даних. Для виконання цього завдання використовуються міграції. Робляться міграції за допомогою почергового введення наступних консольних команд(рис. 3.9).

```
~/project_name$ python manage.py makemigrations app_name
Migrations for 'app_name':
  app_name/migrations/0001_initial.py:
  - Create model Data
~/project_name$ python manage.py migrate app_name
Operations to perform:
  Apply all migrations: app_name
Running migrations:
  Rendering model states... DONE
  Applying app_name.0001_initial... OK
```

Рис. 3.9 Процес зберігання моделі в базі даних

`makemigrations` – фіксує зміни в моделях та створює міграції.

`migrate` – реалізує керування міграціями та переходи між ними.

Після виконання цих двох команд в папці migrations з'явиться новий файл. Тобто міграції дуже схожі на комміти – своєрідний контроль версій для бази даних.

backend.py

Призначення цього файлу – збирання усієї необхідної системної інформації про файли та директорії сервера. Потім отримані дані будуть формуватись у створену нами вище модель та відображатись на сайті. Також цей файл виконує всі основні функції притамані звичайному файловому менеджеру, зокрема створення, копіювання, видалення і т.д. До речі, більш повні версії ключових частин програмного коду можна переглянути в додатку D.

Так як об'єм файлу значний, розглянемо загально лише окремі ключові інструменти, які були використані.

Пакет *os*

Даний модуль пропонує велику кількість функцій для роботи з операційною системою та файлами. Також він незалежний від ОС та може в значній мірі використовуватись на будь-якій з них. Для нашої програми будуть корисними такі засоби:

`os.walk()` – повертає об'єкт-генератор, з якого можна отримати кортежі для кожного каталогу з переданої в аргументі файлової ієрархії.

Кожен кортеж складається з трьох елементів:

1. Повна адреса чергового каталогу у вигляді рядка.
2. Імена підкаталогів першого рівня вкладеності у формі списку для даного каталогу.
3. Окремий список імен файлів даного каталогу.

`os.chdir()` – дозволяє змінити поточну директорію на іншу, шлях до якої передано в аргументі.

`os.path.isdir()` – перевірка чи є вказаний в аргументі шлях директорією.

`os.path.getsize()` – повертає розмір файла чи папки в байтах.

`os.path.exists()` – перевіряє чи існує вказаний шлях.

`os.system()` – виконує передану у вигляді строки системну команду автоматично прописуючи її в консолі.

`os.makedirs()` – створює директорію разом зі всіма проміжними.

`os.remove()` – видаляє весь вказаний в аргументі шлях.

`os.rename()` – змінює назву файлу або директорії вказаної як перший аргумент. У якості нової назви виступає другий аргумент.

Пакет *shutil*

Даний модуль доповнює пакет os своїм набором функцій високого рівня для обробки файлів та директорій.

`shutil.copyfile()` – копіює вміст файлу вказаного як перший аргумент. Цілью є файл вказаний у другому аргументі, який буде автоматично створений якщо не існує, або перезаписаний в іншому випадку.

`shutil.copytree()` – рекурсивно копіює повне дерево директорій з коренем вказаним як перший аргумент. Місцем призначення виступає директорія вказана як другий аргумент. Вона не повинна існувати і буде створена разом з усіма пропущеними батьківськими директоріями.

`shutil.rmtree()` – видаляє вказану у першому аргументі директорію разом з усіма дочірніми директоріями.

`shutil.move()` – рекурсивно переміщує файл або директорію вказаних як перший аргумент. Цілью є шлях вказаний у другому аргументі. Якщо це існуюча директорія, то ресурс переміщується всередину. У випадку з існуючим файлом, він буде перезаписаний.

Папка *templates*

У файлах в папці templates використовуються мова розмітки HTML, CSS та шаблонізатори Django. Саме тут ми формуємо відображення створених нами моделей даних та інших частин сайту, з якими буде взаємодіяти користувач.

Для досягнення комфортного та гармонійного вигляду активно використовувайся найбільш популярний фронт-енд фреймворк Bootstrap.

Bootstrap – це безкоштовний та зручний набір інструментів для розробки сайтів та веб-сервісів. Вже включає в себе готові CSS та HTML-шаблони для оформлення веб-форм, міток, типографіки, кнопок, блоків навігації та інших базових компонентів веб-інтерфейсу, включаючи JavaScript-розширення. На офіційному сайті цього фреймворку можна легко знайти багато різних якісних прикладів коду для стильного оформлення, зокрема використаних і для нашого сайту. Через це ми не будемо детально розглядати сам веб-інтерфейс, зосередивши основну увагу на функціональній роботі та особливостях нашого сайту. За організацію цієї частини сайту відповідає вбудований шаблонізатор Django(рис. 3.10).

```
{% for data in table1 %}
{# table1 - QueryDict #}
       <p>{{ data.name|title }}</p>
       <p>{{ data.type }}</p>
       <p>{{ data.size }}</p>
{% endfor %}
```

Рис. 3.10 Приклад використання шаблонізатора Django

Цей шаблонізатор простий та інтуїтивно зрозумілий, тому що схожий на звичайну мову програмування. Тому уточнення необхідне лише декільком особливим моментам:

1) Змінні виводять значення з контексту, який є словником(QueryDict).

Змінні виділяються `{{` і `}}`, наприклад:

My first name is `{{ first_name }}`. My last name is `{{ last_name }}`.

Якщо словник має вигляд `{ 'first_name': 'John', 'last_name': 'Doe' }` шаблон відрендерить: My first name is John. My last name is Doe.

Звернення до ключів словника, атрибутів об'єктів і елементів списку виконується через точку:

```
{{ My_dict.key }}
{{ My_object.attribute }}
{{ My_list.0 }}
```

2) Теги дозволяють додавати довільну логіку в шаблон. Наприклад, теги можуть виводити текст, додавати логічні оператори, такі як "if" або "for", отримувати вміст з бази даних, або надавати доступ до інших тегами.

Теги виділяються `{%` і `%}`, наприклад:

```
{% If user.is_authenticated %}
    Hello, {{ user.username }}.
{% Endif %}
```

3) Фільтри перетворюють змінні і аргументи тегів згідно вказаному стилю.

Можуть виглядати таким чином:

```
{{ Django | title }}
```

Для контексту `{ 'django': 'the web framework' }` цей шаблон виведе:

The Web Framework

Деякі фільтри приймають аргументи:

```
{{ Today | date: "Y-m-d" }}
```

4) Коментарі

Коментарі виглядають таким чином:

```
{# This will not be rendered #}
```

Тег `{% comment %}` дозволяє створювати багаторядкові коментарі.

Висновки до розділу 3

В результаті виконання поточного розділу бакалаврської роботи направленого на формування архітектури програмного продукту можна чітко побачити звичну для проектів фреймворку Django структуру. Таким чином, продукт відповідає всім необхідним та звичним правилам побудови веб-сервісів. Інші розробники подібного програмного забезпечення легко та швидко зможуть розібратись в архітектурі даного продукту.

Також, при написанні коду були використані сучасні швидкодіючі пакети та бібліотеки мови програмування Python. Активно застосовувались різноманітні вбудовані інструменти Django, дотримуючись правил та зберігаючи звичну архітектуру. Було зручно організовано використання CSS та HTML-шаблонів, що надало можливість швидко змінювати та корегувати інтерфейс та зовнішній вигляд веб-додатку.

РОЗДІЛ 4

ОГЛЯД РОЗРОБЛЕНОГО ПРОДУКТУ

4.1 Демонстрація роботи продукту

В цьому підрозділі ми зробимо огляд створеного продукту, оцінимо його можливості та ефективність, розробимо інструкцію по використанню для клієнта.

При відкритті сайту користувач одразу ж бачить ілюстрацію як на рис. 4.1.

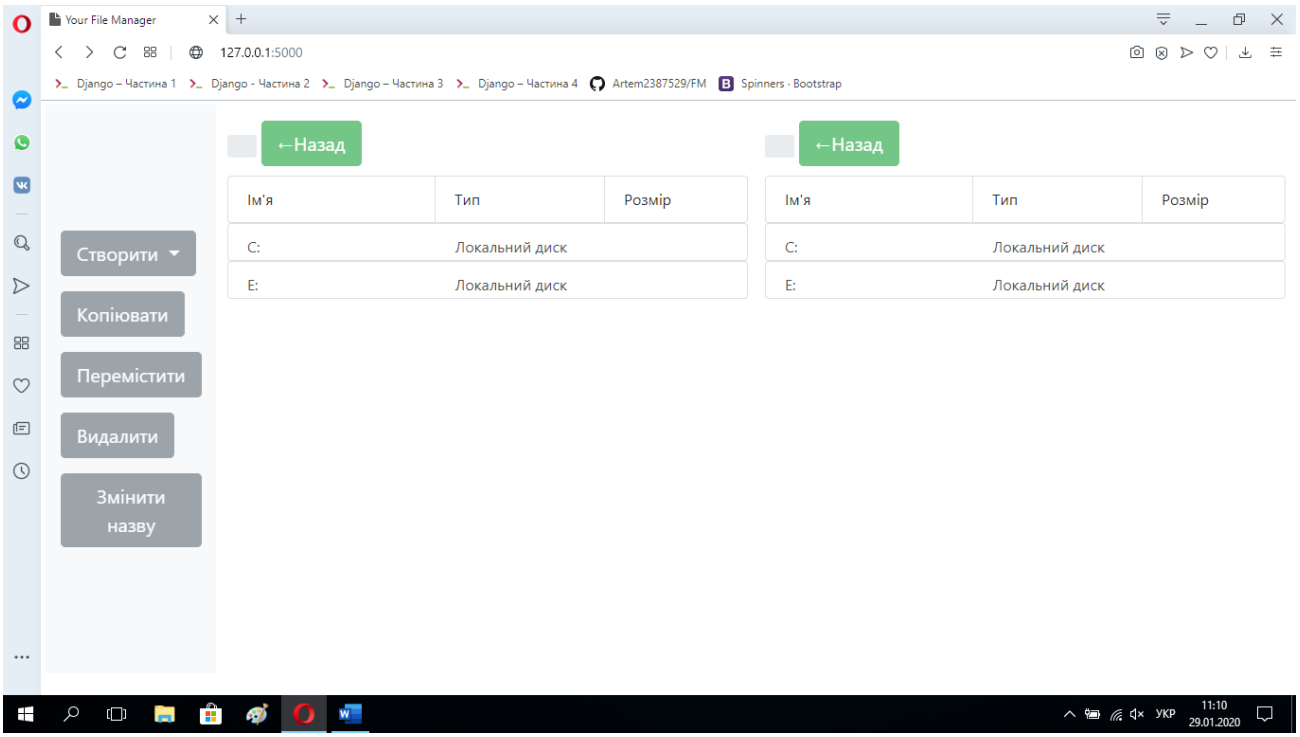


Рис. 4.1 Зовнішній вигляд головної сторінки сайту

Як і в багатьох звичайних файлових менеджерів (наприклад Total Commander), ключову частину сайту складають дві таблички. Вони відображають звичні в таких випадках дані про наші папки, файли, диски: ім’я, тип та розмір(рис. 4.2).

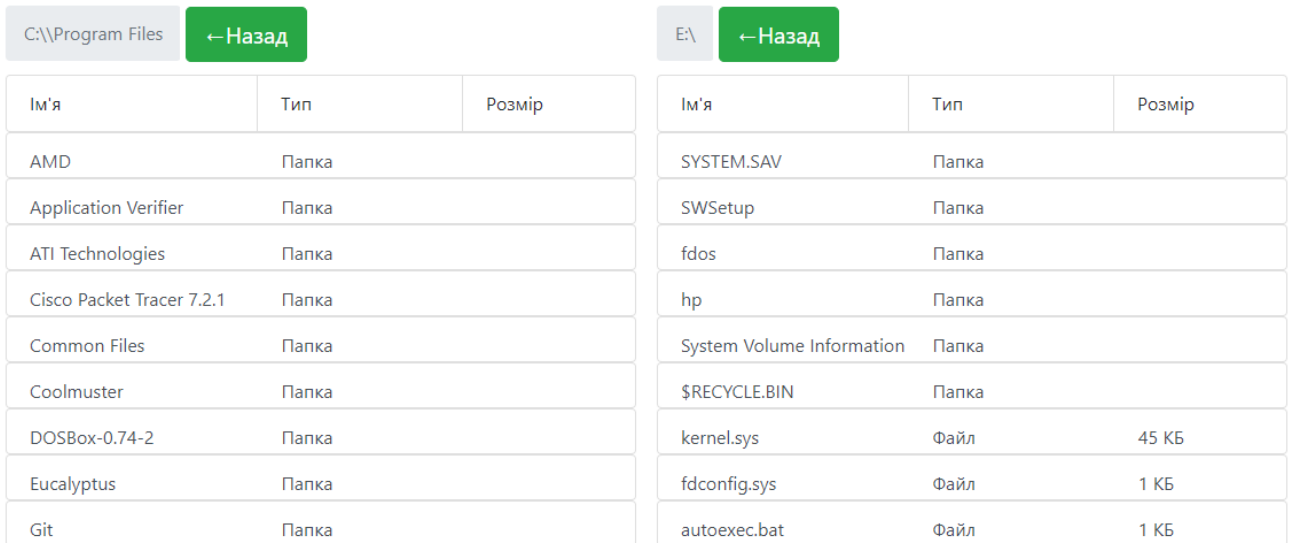


Рис. 4.2 Таблиці ієрархій даних сервера

Нагадаю, що ця інформація формується в файлі backend.py, перетворюється у моделі та виводиться на сайті.

Переходи по папках здійснюється за допомогою натискання на необхідну користувачу папку, повернутись в попередню директорію можна за допомогою кнопки ‘‘Назад’’. Поряд з нею також можна відслідковувати повний шлях до поточної директорії. Натискання на файл призводить до запуску цього файлу.

Також в лівій частині сайту можна спостерігати своєрідну панель управління. Тут знаходяться кнопки з назвами: ‘‘Створити’’, ‘‘Копіювати’’, ‘‘Перемістити’’, ‘‘Видалити’’ та ‘‘Змінити назву’’(рис. 4.3). Гадаю призначення цих кнопок очевидне, тому ми просто продемонструємо і розберемо принципи їх роботи та використання. Почнемо по порядку з кнопки ‘‘Створити’’. Після першого натискання відкривається додаткове спливаюче вікно(рис. 4.4).

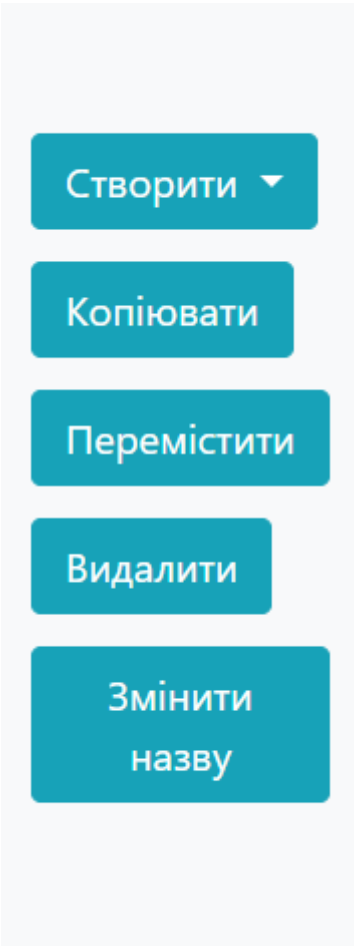


Рис. 4.3 ‘‘Панель управління’’

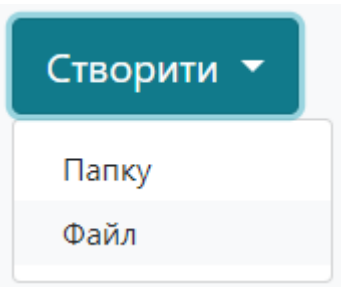


Рис. 4.4 Меню кнопки ‘‘Створити’’

Зрозуміло, що в цьому меню користувачу необхідно обрати, що саме він хоче створити: файл чи папку. Якщо користувач обирає створення папки, перед ним з’являється наступне вікно(рис. 4.5).

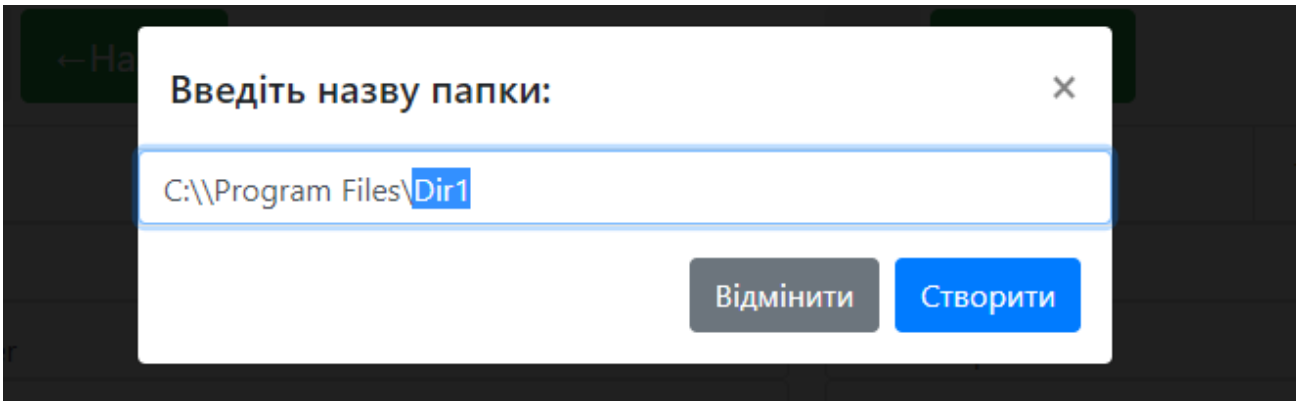


Рис. 4.5 Вікно створення папки

Тут ми маємо змогу спостерігати рядок, в якому вже наявний шлях до поточної директорії з лівої таблички. Нам залишається лише дописати назву нової папки або ієрархію папок, яка і буде створена тут. Якщо нам потрібно створити папку в іншому місці, необхідно прописати повний шлях до нього. Детальний розбір обробки форм та запитів користувача можна подивитись у вигляді алгоритму у додатку С.

Після натискання кнопки ‘‘Створити’’ ми автоматично переходимо в новостворену папку(рис. 4.6).

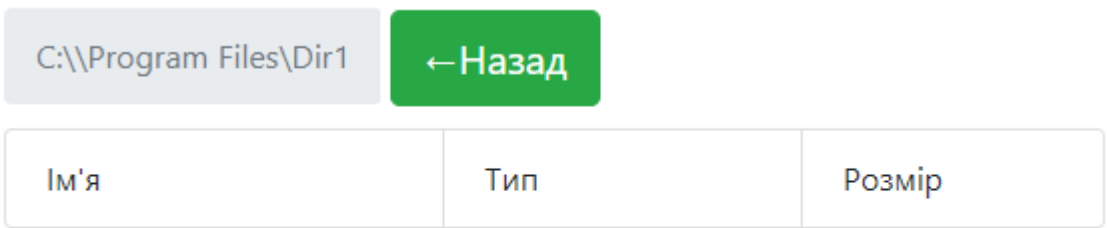


Рис. 4.6 Вигляд таблиці після створення нової папки Dir1

Створення файлів робиться по аналогії та майже нічим не відрізняється від створення папок(рис. 4.7).

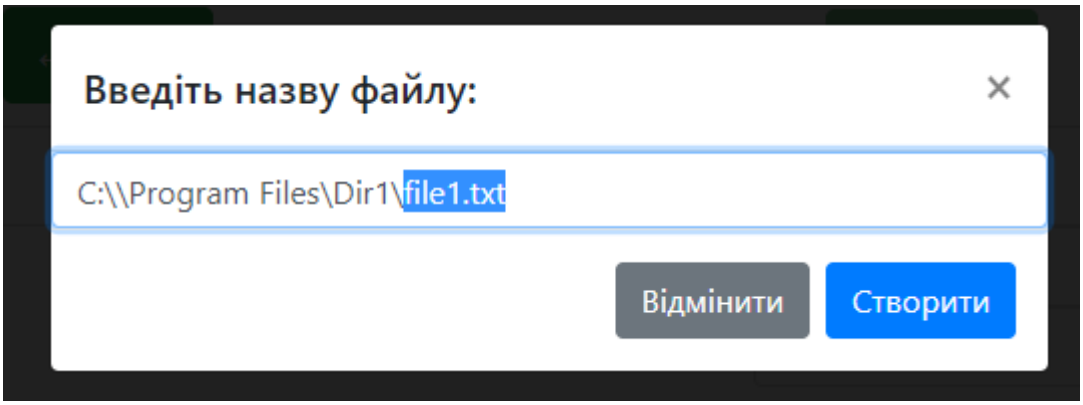


Рис. 4.7 Вікно створення файлу

Єдина відмінність – після натискання кнопки ‘‘Створити’’ ми автоматично переходимо в папку, де був створений наш файл(рис. 4.8).

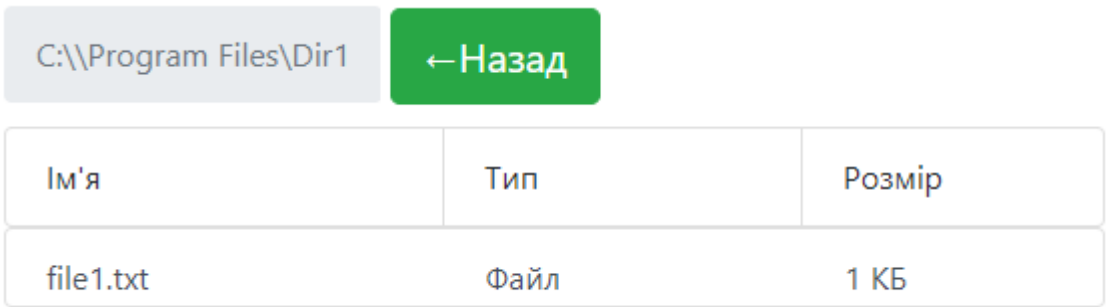


Рис. 4.8 Вигляд таблиці після створення нового файлу file1.txt в папці Dir1

Перейдемо до наступного інструменту звичного для всіх файлових менеджерів, який реалізовано у вигляді кнопки ‘‘Копіювати’’. Після натискання на неї перед користувачем з’являється вікно, показане на рис. 4.9. В верхній частині вікна відображаються шляхи, що відповідають шляхам до поточних директорій в лівій та правій таблиці відповідно. Тобто для зручності користування перед копіюванням файлів можна наперед знайти та переглянути папку призначення. В той же час у разі потреби цей шлях можна змінити на інший прямо у вікні для копіювання. Далі потрібно відмітити галочкою всі потрібні файли та папки та натиснути кнопку ‘‘Копіювати’’.

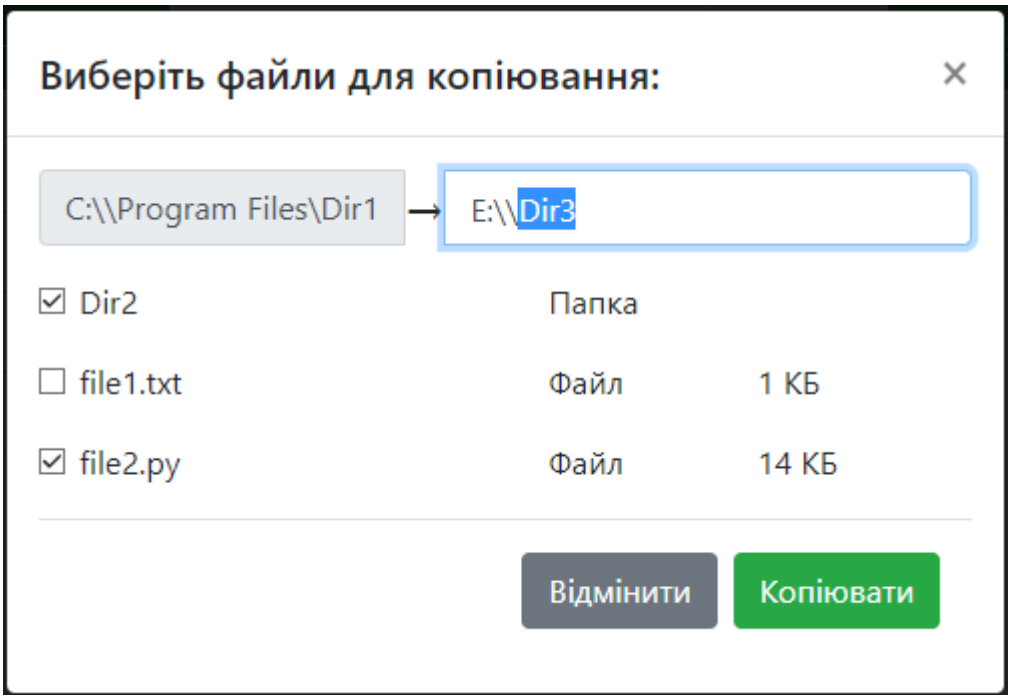


Рис 4.9 Вікно копіювання

Після цього сторінка сайту автоматично оновиться(рис. 4.10).

Рис. 4.10 Папки після копіювання

Робота з наступною кнопкою ‘‘Перемістити’’ з нашої панелі керування аналогічна до кнопки ‘‘Копіювати’’(рис. 4.11).

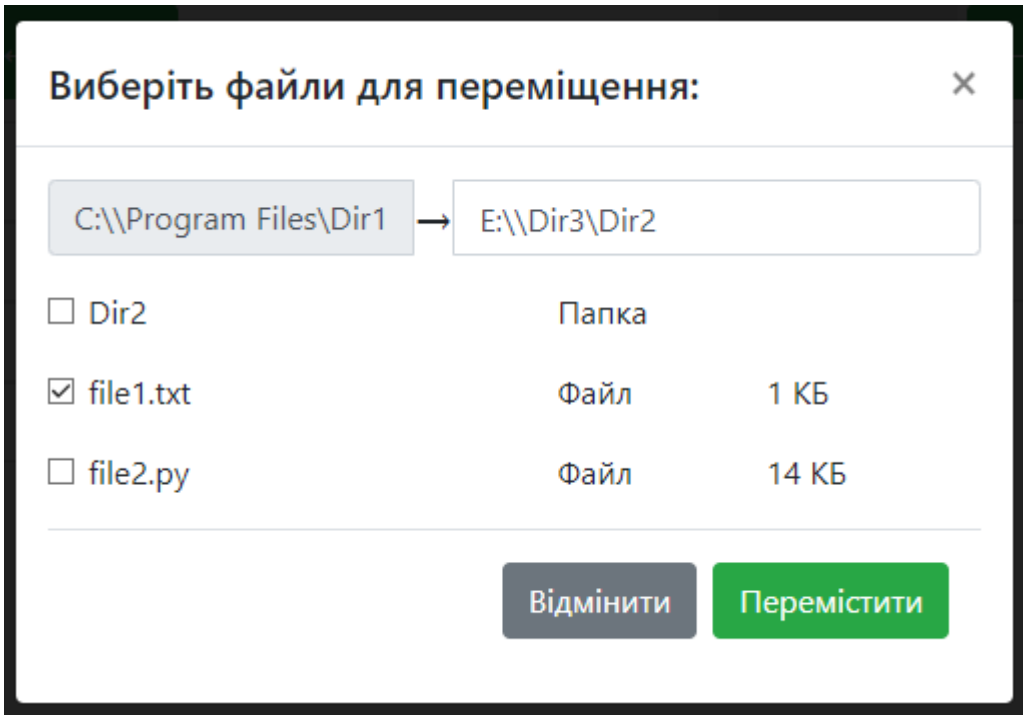


Рис. 4.11 Вікно переміщення

Лише результат виконання очікувано буде інакшим(рис. 4.12).

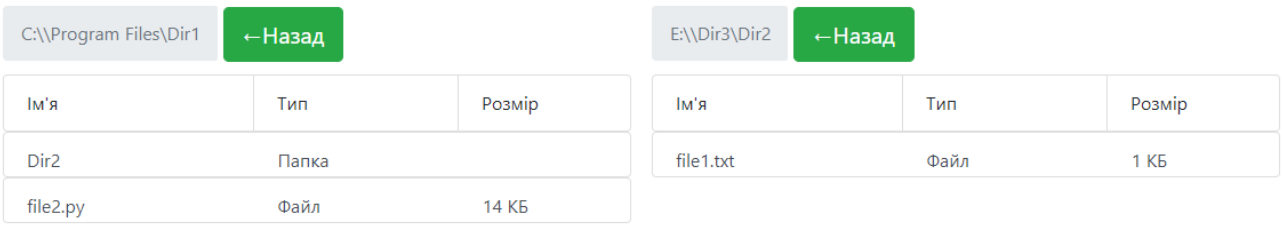


Рис. 4.12 Папки після переміщення

Наступна важлива кнопка в нашому списку – це ‘‘Видалити’’. Після натискання на неї спливає наступне вікно(рис. 4.13).

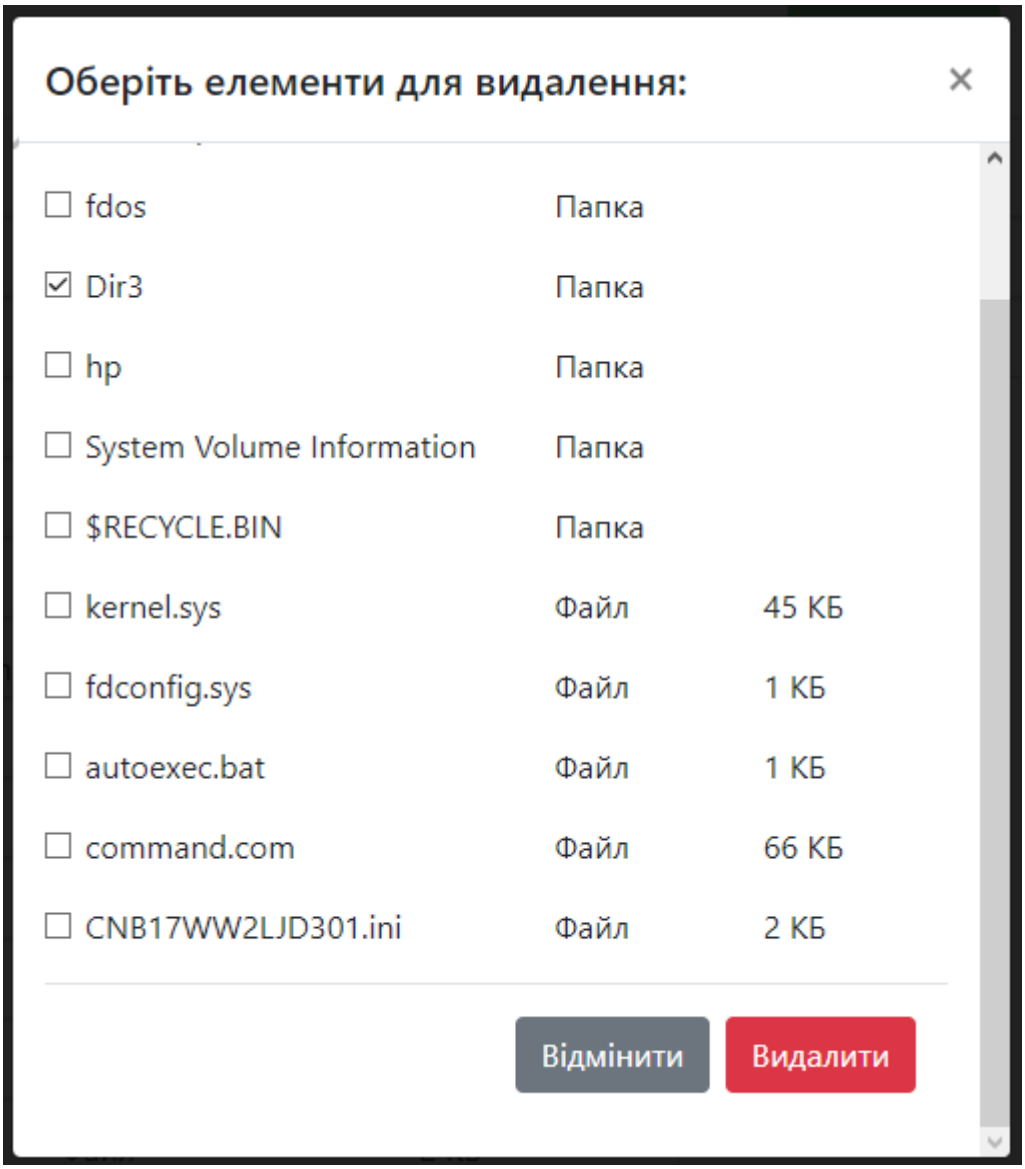


Рис. 4.13 Вікно видалення файлів та папок

Тут відображається список всіх файлів та папок з лівої таблиці, з яких ми галочками обираємо претендентів на видалення. Після цього ми натискаємо на кнопку ‘‘Видалити’’, відбувається видалення після чого сторінка автоматично оновлюється(рис 4.14).

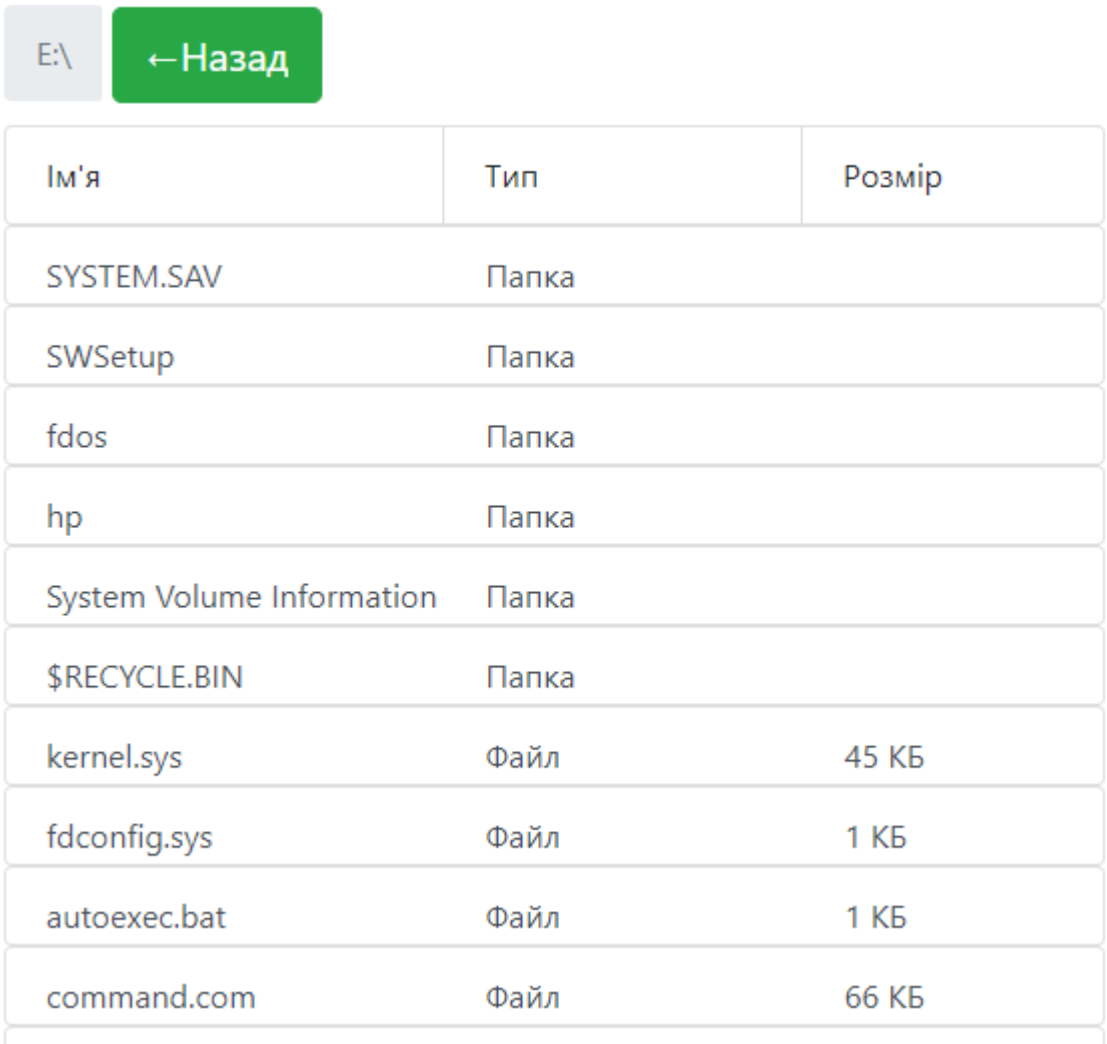


Рис. 4.14 Таблиця після видалення папки Dir3

Останньою по порядку але не по значенню йде кнопка ‘‘Змінити назву’’ (рис. 4.15). На відміну від інших кнопок, які просто виконували звичні для більшості файлових менеджерів функції, реалізація цієї кнопки має значне покращення. Своєрідною її ‘‘фішкою’’ є можливість одночасної зміни назв одразу всіх файлів присутніх в поточній директорії(рис. 4.16).

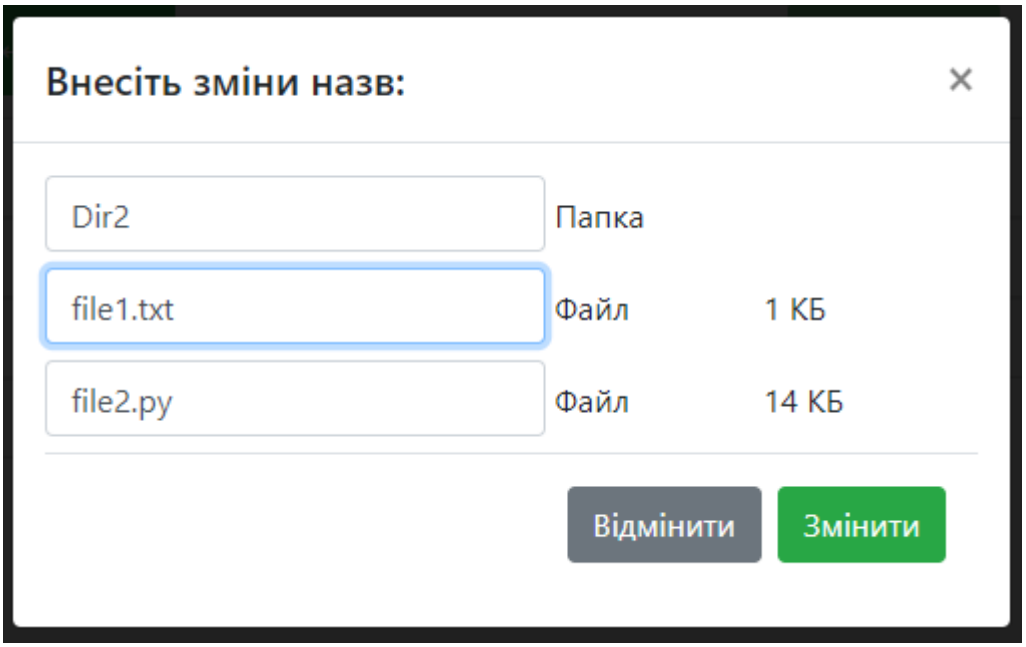


Рис. 4.15 Вікно для зміни назв файлів та папок

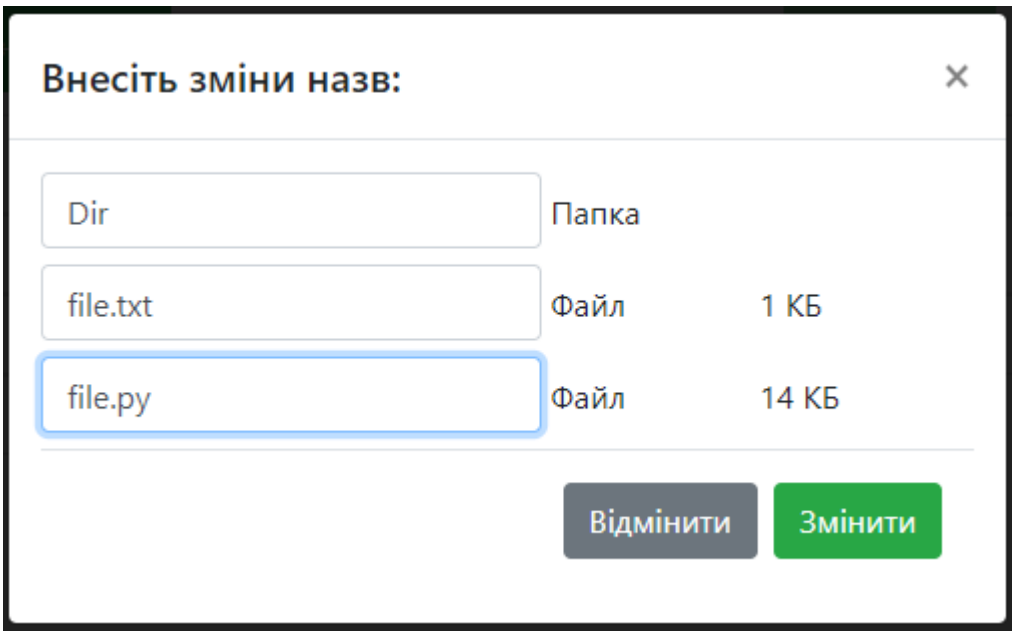


Рис. 4.16 Демонстрація особливості роботи кнопки ‘‘Змінити назву’’

Після натискання кнопки ‘‘Змінити’’, відбувається виконання усіх внесених змін, сторінка знов швидко оновлюється та демонструє необхідний результат(рис 4.17).

Рис. 4.17 Результат виконання зміни назв

Висновки до розділу 4

Створений програмний продукт має приємний, добре організований та звичний для файлових менеджерів зовнішній вигляд.

На основі демонстрації вбудованих можливостей можна зробити висновок о виконанні всіх необхідних умов для зручного та швидкого користування створеним файловим менеджером в інтернет-середовищі. Це в свою чергу відкриває значні шляхи для розвитку інтернет-технологій, адже в такому разі користувачам немає потреби зберігати великі ієрархії директорій та файлів на своєму персональному комп’ютері. Достатньо лише підтримувати зв’язок з ними через пристрій з підключенням до Інтернету, в той час як спеціальні компанії можуть забезпечувати ваші дані захистом, резервними копіями, а сервер буде мати значно потужніші та швидші компоненти, що будуть виконувати роль звичного для нас системного блоку.

ВИСНОВКИ

У результаті виконання даного дипломного проекту було розроблено файловий менеджер, що реалізує всі необхідні інструменти та засоби та призначений для налаштування та організації даних віддаленого сервера через інтернет-середовище. Аналізуючи наявні рішення було вирішено, що оптимальним стеком засобів реалізації системи буде мова програмування Python в поєднанні з фреймворками Django та Bootstrap. Продукт має вигляд сучасного та зручного веб-сервісу, адже для його створення було використано найпопулярніші та постійно оновлювальні інструменти.

Також у ході підготовки до роботи було досліджено останні новини та стан сфери інтернет-технологій, її значний прогрес та розвиток, підтверджено актуальність розробки та необхідність подальшії роботи в цьому напрямку.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Марк Лутц, Изучаем Python [Текст]. – O'Reilly. – 2011. – 720 с.
2. Марк Лутц, Программирование на Python, том 1 [Текст]. – O'Reilly. – 2011. – 992 с.
3. Марк Лутц, Python. Кишеньковий довідник [Текст]. – O'Reilly. – 2015. – 320 с.
4. Total Commander – офіційна сторінка [Електронний ресурс]. – Режим доступу: https://www.ghisler.com/. – Дата доступу: грудень 2019.
5. Файлові менеджери. Що це та якими вони бувають [Електронний ресурс]. – Режим доступу: https://lickeys.ru/uk/programmy/kakie-failovye-menedzhery-byvayut-failovye-menedzhery/. – Дата доступу: грудень 2019.
6. Python. Підручники та корисні посилання [Електронний ресурс]. – Режим доступу: https://ru.wikibooks.org/wiki/Python/. – Дата доступу: січень 2019.
7. Майк МакГрат, Python [Текст]. – ООО Издательство Эксмо. – 2015. – 192 с.
8. Фреймворк Bootstrap [Електронний ресурс]. – Режим доступу: https://getbootstrap.com. – Дата доступу: лютий 2020.
9. Фреймворк Django. Documentation [Електронний ресурс]. – Режим доступу: https://docs.djangoproject.com/en/3.0/. – Дата доступу: лютий 2020.
10. Інформація про Django [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Django/. – Дата доступу: березень 2020.
11. Веб-фреймворк Django [Електронний ресурс]. – Режим доступу: https://developer.mozilla.org/ru/docs/Learn/Server-side/Django/. – Дата доступу: березень 2020.
12. Інформація про Total Commander [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Total_Commander/. – Дата доступу: квітень 2020.

13. Переваги Інтернету [Електронний ресурс]. – Режим доступу: https://sites.google.com/site/perevagitanedolikiinternetu/home/perevagi-internetu-1/. – Дата доступу: квітень 2020.
14. Інтернет-середовище [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Інтернет/. – Дата доступу: травень 2020.
15. Файловий менеджер Frigate – офіційна сторінка [Електронний ресурс]. – Режим доступу: http://www.frigate3.com/. – Дата доступу: травень 2020.
16. Николай Прохоренок, Владимир Дронов, Python 3 и PyQt 5. Разработка приложений [Текст]. – BHV. – 2016. – 832 с.

Додатки

Додаток А

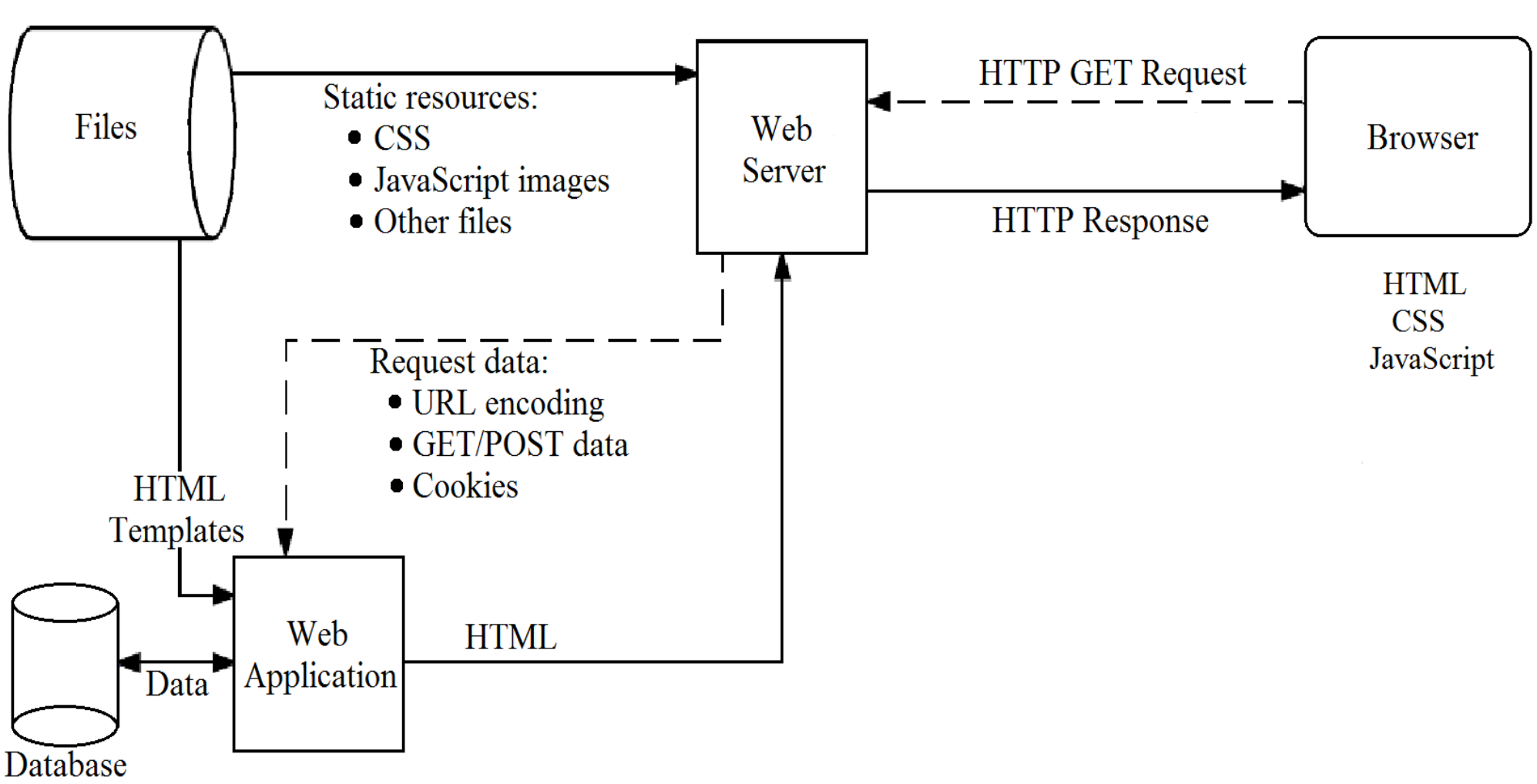
Структурна схема. Зв’язок між вузлами

Аркушів 1

Київ 2020

Server-side

Client-side



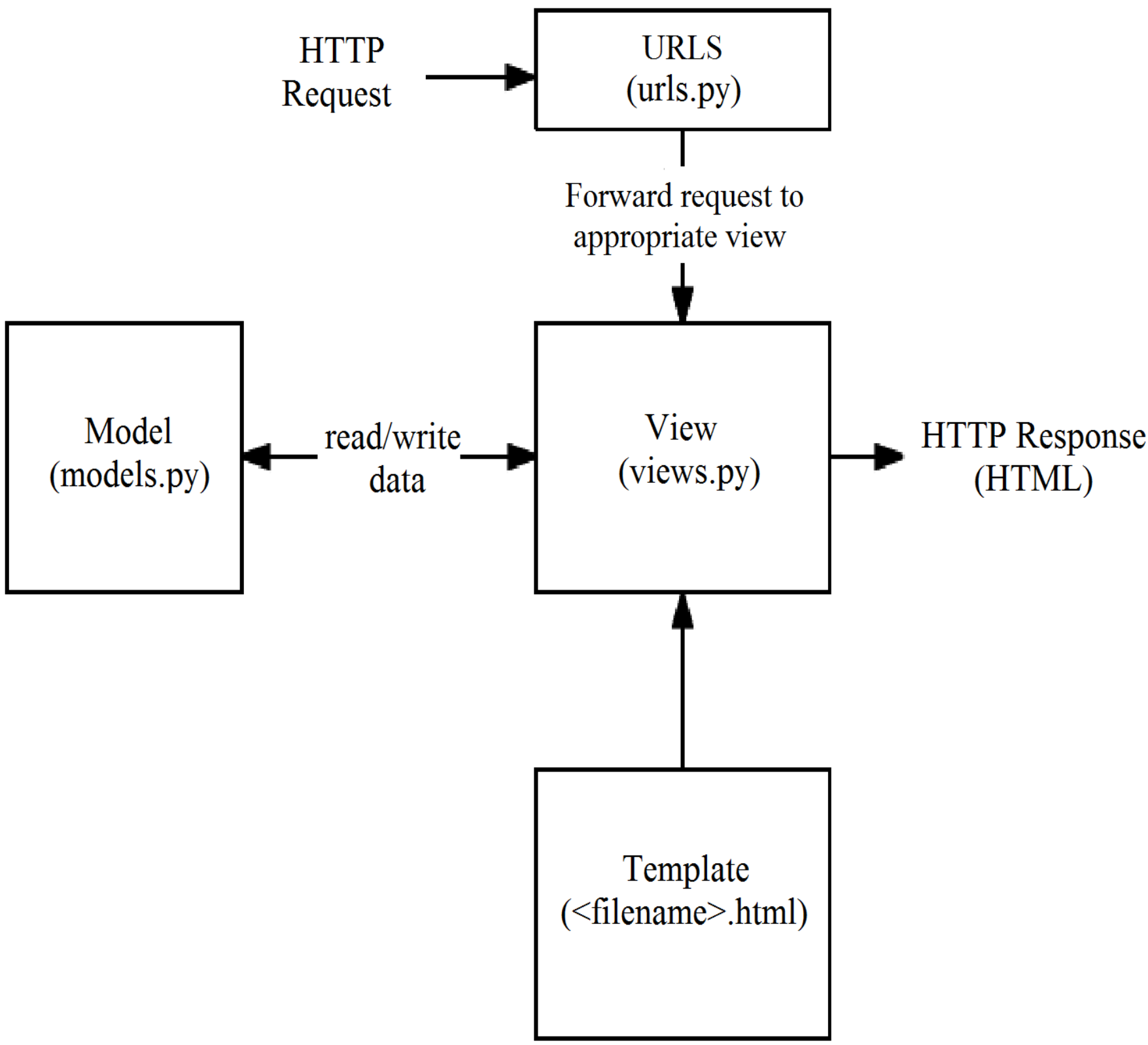
					ІАЛЦ.045440.004 Д1		
Зм.	Арк.	№ докум.	Підпис	Дата			
Розробив		Палієнко А. П.			Структурна схема. Зв'язок між вузлами	Лит.	Аркуш 1 / Аркушів 1
Перевірив		Антонюк А. І.					
Т.Контр.						НТУУ «КПІ», ФІОТ, гр. ІО-62	
Н.Контр.		Сімоненко В.П.					
Затвердив		Стіренко С.Г.					

Додаток В

Функціональна схема. Взаємодія модулів програми

Аркушів 1

Київ 2020



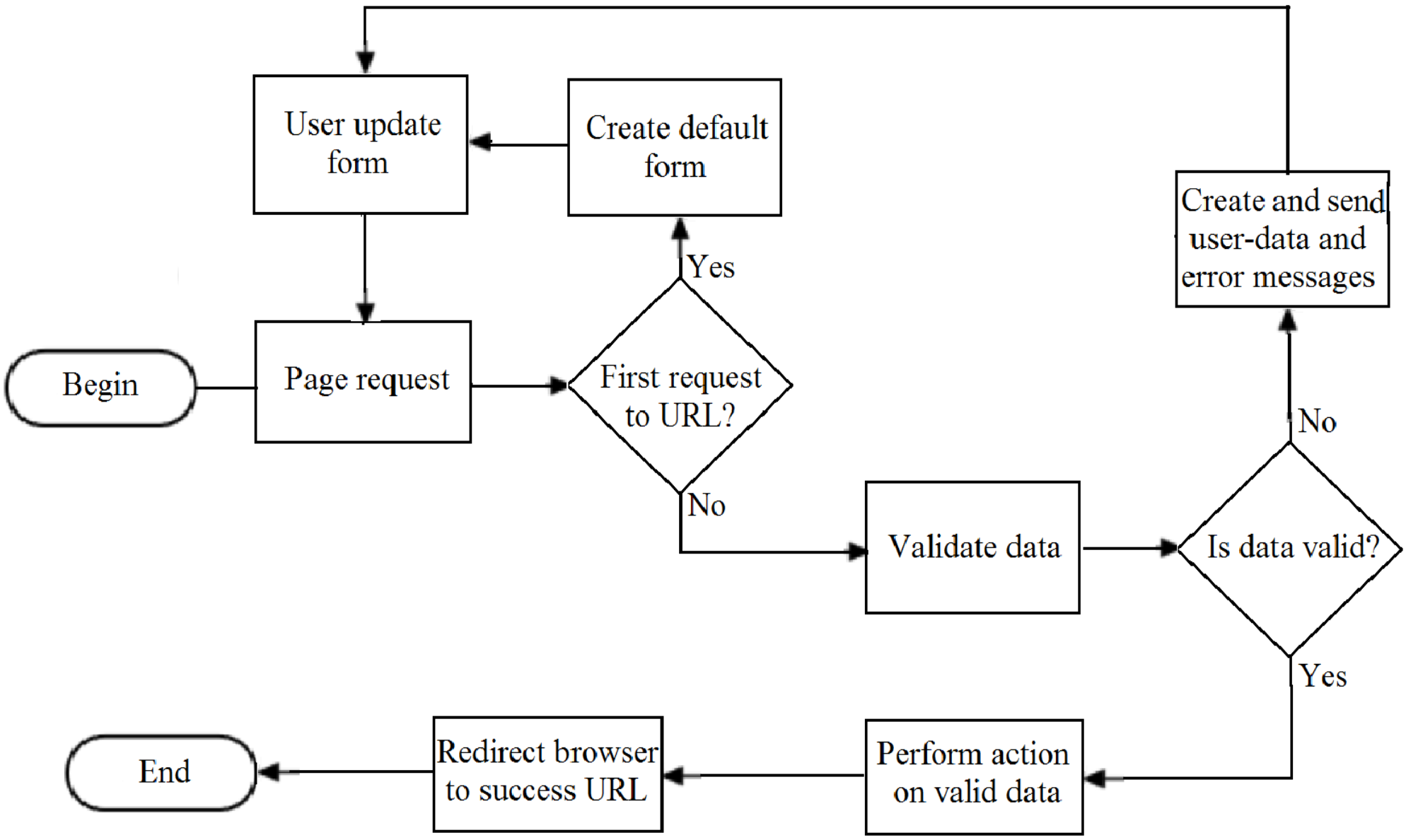
Зм.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.045440.005 Д2		
Розробив		Палієнко А. П.			Функціональна схема. Взаємодія модулів програми	Лит.	Аркуш 1 / Аркушів 1
Перевірив		Антонюк А. І.					
Т.Контр.						НТУУ «КПІ», ФІОТ, гр. ІО-62	
Н.Контр.		Сімоненко В.П.					
Затвердив		Стіренко С.Г.					

Додаток С

Принципова схема. Алгоритм обробки запитів

Аркушів 1

Київ 2020



Зм.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.045440.006 Д3		
Розробив		Палієнко А. П.			Принципова схема. Алгоритм обробки запитів	Лит.	Аркуш 1 / Аркушів 1
Перевірив		Антонюк А. І.					
Т.Контр.						НТУУ «КПІ», ФІОТ, гр. ІО-62	
Н.Контр.		Сімоненко В.П.					
Затвердив		Стіренко С.Г.					

Додаток D

Ключові елементи коду програми

Аркушів 11

Київ 2020

urls.py

```
from django.urls import path
from .views import *

urlpatterns = [
    path('', main_page, name='main_page'),
    path('delete/', delete_thing.as_view(), name='delete_thing'),
    path('change/', change_thing.as_view(), name='change_thing'),
    path('move/', move_thing.as_view(), name='move_thing'),
    path('copy/', copy_thing.as_view(), name='copy_thing'),
    path('create_dir/', create_dir.as_view(), name='create_dir'),
    path('create_file/', create_file.as_view(), name='create_file'),
    path('<str:slug>/', click_thing.as_view(), name='click_thing'),
]
```

views.py

```
from django.shortcuts import render
from .models import Data
from .backend import backend, open_file
from django.views.generic import View
import os
from shutil import rmtree, move, copytree, copyfile

class st:
    savetable1 = {}
    savetable2 = {}
    obj1 = ''
    obj2 = ''


def new_table(path=''):
    table = Data.objects.all()
    for h in range(len(table)):
        obj = Data.objects.get(path=table[h].path)
        obj.delete()
    if path!='':
        if path[-1]!='/' and path[-1]!='\\':
            listofthings = backend(path)
        elif path=="C:\\":
            listofthings = backend(path)
        else:
            listofthings = backend(path[:-1])
    else:
        listofthings = backend(path)
    for i in range(len(listofthings)):
        try:
            Data.objects.create(
                path=listofthings[i].path,
                name=listofthings[i].name,
                type=listofthings[i].type,
                size=listofthings[i].size)
        except: pass
    table = Data.objects.all()
    return table


def main_page(request):
    st.savetable1 = new_table()
    st.savetable2 = new_table()
    class Thi():
```

```
        def __init__(self):
            self.name = ""
            self.path = ""
            self.slug = 'obj'
        def __str__(self):
            return self.name
    st.obj1 = Thi()
    st.obj2 = Thi()
    return render(request, 'fm.html', {'obj1': st.obj1, 'obj2': st.obj2,
'table1': st.savetable1, 'table2': st.savetable2})


class create_dir(View):
    def post(self, request):
        p = request.POST.get('fullpath')
        if not os.path.exists(p):
            os.makedirs(p)
        buf, n = '', ''
        for j in p[::-1]:
            if j == '\\' or j == '/':
                break
            else:
                buf += j
        for k in buf[::-1]:
            n += k
        class Thin:
            def __init__(self):
                self.name = n
                self.path = p
                self.slug = 'obj'
            def __str__(self):
                return self.name
        st.savetable1 = new_table(p)
        st.obj1 = Thin()
        return render(request, 'return_fm.html', {'obj1': st.obj1, 'obj2':
st.obj2, 'table1': st.savetable1,
                                                  'table2': st.savetable2,
'numbtable': request.GET.get("numbtable")})


class create_file(View):
    def post(self, request):
        p = request.POST.get('fullpath')
        buf, n = '', ''
        for j in p[::-1]:
            if j == '\\' or j == '/':
                break
            else:
                buf += j
        for k in buf[::-1]:
            n += k
        open(p, "a+")
        class Thin:
            def __init__(self):
                self.name = n
                self.path = p[:-(len(n)+1)]
                self.slug = 'obj'
            def __str__(self):
                return self.name
        st.savetable1 = new_table(p[:-(len(n)+1)])
        st.obj1 = Thin()
        return render(request, 'return_fm.html', {'obj1': st.obj1, 'obj2':
st.obj2, 'table1': st.savetable1,
                                                  'table2': st.savetable2,
```

```
'numbtable': request.GET.get("numbtable")})


class delete_thing(View):
    def post(self, request):
        p = request.POST.get('path')[:-(len(request.POST.get('name'))+1)]
        t = dict(request.POST)
        delete_list = t.get('file')
        for i in delete_list:
            try: os.remove(p+"\\"+i)
            except: rmtree(p+"\\"+i)
        buf = ''
        for i in p:
            if i != '/' and i != "\\":
                buf += i
            else:
                buf = ''
        class Thin:
            def __init__(self):
                if buf == "":
                    self.name = p[:2]
                else:
                    self.name = buf
                self.path = p
                self.slug = 'obj'
            def __str__(self):
                return self.name
        st.savetable1 = new_table(p)
        st.obj1 = Thin()
        return render(request, 'return_fm.html', {'obj1': st.obj1, 'obj2':
st.obj2, 'table1': st.savetable1,
                                                  'table2': st.savetable2,
'numbtable': request.GET.get("numbtable")})


class change_thing(View):
    def post(self, request):
        p = request.POST.get('path')[:-(len(request.POST.get('name'))+1)]
        t = dict(request.POST)
        old_list = t.get('name')
        new_list = t.get('newname')
        for n in range(len(new_list)):
            if new_list[n] != old_list[n]:
                os.rename(p+"\\"+old_list[n], p+"\\"+new_list[n])
        buf = ''
        for i in p:
            if i != '/' and i != "\\":
                buf += i
            else:
                buf = ''
        class Thin:
            def __init__(self):
                self.name = buf
                self.path = p
                self.slug = 'obj'
            def __str__(self):
                return self.name
        st.savetable1 = new_table(p)
        st.obj1 = Thin()
        return render(request, 'return_fm.html', {'obj1': st.obj1, 'obj2':
st.obj2, 'table1': st.savetable1,
                                                  'table2': st.savetable2,
'numbtable': request.GET.get("numbtable")})
```

```
class move_thing(View):
    def post(self, request):
        moveto = request.POST.get('path')
        path = request.POST.get('path1')
        t = dict(request.POST)
        files = t.get('file')
        try:
            for f in files:
                src = path+"\\"+f
                dst = moveto+"\\"+f
                move(src, dst)
        except:
            pass
        x = new_table(moveto)
        for r in x:
            pass
        st.savetable2 = x
        for rq in st.savetable2:
            pass
        buf = ''
        for i in path:
            if i != '/' and i != "\\":
                buf += i
            else:
                buf = ''
        class Thin:
            def __init__(self):
                self.name = buf
                self.path = path
                self.slug = 'obj'
            def __str__(self):
                return self.name
        st.obj1 = Thin()
        st.savetable1 = new_table(path)
        for rr in st.savetable1:
            pass
        return render(request, 'return_fm.html', {'obj1': st.obj1, 'obj2':
st.obj2, 'table1': st.savetable1,
                                                  'table2': st.savetable2,
'numbtable': request.GET.get("numbtable")})


class copy_thing(View):
    def post(self, request):
        moveto = request.POST.get('path')
        path = request.POST.get('path1')
        t = dict(request.POST)
        files = t.get('file')
        try:
            for f in files:
                src = path+"\\"+f
                dst = moveto+"\\"+f
                try:
                    copyfile(src, dst)
                except:
                    copytree(src, dst)
        except:
            pass
        buf = ''
        for i in moveto:
            if i != '/' and i != "\\":
                buf += i
            else:
```

```
                buf = ''
        class Thin:
            def __init__(self):
                self.name = buf
                self.path = moveto
                self.slug = 'obj'
            def __str__(self):
                return self.name
        st.obj2 = Thin()
        st.savetable2 = new_table(moveto)
        return render(request, 'return_fm.html', {'obj1': st.obj1, 'obj2':
st.obj2, 'table1': st.savetable1, 'table2': st.savetable2, 'numbtable':
request.GET.get("numbtable")})
```

models.py

```
from django.db import models
from django.utils.text import slugify


def gen_slug(s):
    new_slug = slugify(s, allow_unicode=True)
    return new_slug


class Data(models.Model):
    slug = models.SlugField(max_length=150, blank=True, unique=True)
    path = models.TextField(blank=True)
    name = models.TextField(blank=True)
    type = models.TextField(blank=True)
    size = models.TextField(blank=True)

    def save(self, *args, **kwargs):
        if not self.id:
            self.slug = gen_slug(self.path)
        super().save(*args, **kwargs)

    def __str__(self):
        return self.name
```

backend.py

```
import os


def backend(path):
    if path == '':
        treec = os.walk('C:/')
        treed = os.walk('D:/')
        treee = os.walk('E:/')

        class Thing:
            def __init__(self, path):
                buf, name = '', ''
                for j in path[::-1]:
                    if j == '\\' or j == '/':
                        break
                    else:
                        buf += j
                for k in buf[::-1]:
                    name += k
```

```
            if len(name) == 0:
                name = path[:-1]
            self.name = name
            self.path = self.name + "\\"
            if len(name) < 3:
                self.type = 'Локальний диск'
                self.size = ''
            elif os.path.isdir(path) == True:
                self.type = 'Папка'
                self.size = ''
            else:
                self.type = 'Файл'
                os.chdir(path[:-len(name)])
                buf = "%14d" % (os.path.getsize(name) / 2 ** 10)
                size = buf + ' КБ'
                self.size = size

    def localdisks(tree):
        for i in tree:
            disk = Thing(str(i[0])[:-1])
            listofdisks.append(disk)
            break

    listofdisks = []

    localdisks(treec)
    localdisks(treed)
    localdisks(treee)
    return listofdisks

else:

    if len(path) == 1:
        path += ":\\"

    tree = os.walk(path)

    class Thing:
        def __init__(self, path):
            buf, name = '', ''
            for j in path[::-1]:
                if j == '\\' or j == '/':
                    break
                else:
                    buf += j
            for k in buf[::-1]:
                name += k
            if len(name) == 0:
                name = path[:-1]
            self.name = name
            self.path = path
            if len(path) < 4:
                self.type = 'Локальний диск'
                self.size = ''
            elif os.path.isdir(path) == True:
                self.type = 'Папка'
                self.size = ''
            else:
                self.type = 'Файл'
                os.chdir(path[:-len(name)])
                buf = "%14d" % (os.path.getsize(name) / 2 ** 10 + 1)
                size = buf + ' КБ'
                self.size = size
```

```
        def findthings(te):
            try:
                if path == "C:\\":
                    for tree in te:
                        for m in tree[1] + tree[2]:
                            if tree[0][-1] != '\\' or tree[0][-1] != '/':
                                dirs = Thing(str(tree[0] + '\\' + m))
                            else:
                                dirs = Thing(str(tree[0] + m))
                            listofthings.append(dirs)
                        break
                    return listofthings
                elif tuple(te)[0][0] == "E:":
                    tre = list(os.walk("E:\\"))
                    for y in tre:
                        for m in y[1] + y[2]:
                            if y[0][-1] != '\\' or y[0][-1] != '/':
                                dirs = Thing(str(y[0] + '\\' + m))
                            else:
                                dirs = Thing(str(y[0] + m))
                            listofthings.append(dirs)
                        break
                    return listofthings
                else:
                    te = os.walk(path)
                    tree = tuple(te)[0]
                    for m in tree[1] + tree[2]:
                        if tree[0][-1] != '\\' or tree[0][-1] != '/':
                            dirs = Thing(str(tree[0] + '\\' + m))
                        else:
                            dirs = Thing(str(tree[0] + m))
                        listofthings.append(dirs)
                    return listofthings
            except:
                te = os.walk(path)
                tree = tuple(te)[0]
                for m in tree[1] + tree[2]:
                    if tree[0][-1] != '\\' or tree[0][-1] != '/':
                        dirs = Thing(str(tree[0] + '\\' + m))
                    else:
                        dirs = Thing(str(tree[0] + m))
                    listofthings.append(dirs)
                return listofthings

        listofthings = []
        l = findthings(tree)
        return l


def open_file(path, name):
    os.system("cd " + path)
    os.system("start " + name)
```

fm.html

```
{% extends 'base.html' %}

{% block return %}
<nav class="navbar navbar-expand-lg-vertical navbar-light bg-light"
     style="width: 180px; position: fixed; left: 5px; height: 600px">
    <ul class="navbar-nav">
        <li class="nav-item dropdown">
```

```
            <button disabled class="btn btn-secondary btn-lg dropdown-toggle"
role="button"
                    data-toggle="dropdown" aria-haspopup="true" aria-
expanded="false">
                Створити
            </button>
        </li>
        <p></p>
        <li class="nav-item">
            <button disabled class="btn btn-secondary btn-
lg">Копіювати</button>
        </li>
        <p></p>
        <li class="nav-item">
            <button disabled class="btn btn-secondary btn-
lg">Перемістити</button>
        </li>
        <p></p>
        <li class="nav-item">
            <button disabled class="btn btn-secondary btn-
lg">Видалити</button>
        </li>
        <p></p>
        <li class="nav-item">
            <button disabled class="btn btn-secondary btn-lg">Змінити
назву</button>
        </li>
    </ul>
</nav>
{% endblock %}

{% block content %}

<div class="container mr-1 mt-1">
    <div class="row" style="top: 20px; left: 100px">
        <div class="col-sm-6" style="position: fixed; overflow: hidden;
height: 600px; width: 600px">
            <div style="overflow-y: auto; height: 600px; width: 5050px">
                {% if obj1 != '' and obj1.name != '' %}
                <div class="list-group" style="height: 75px;">
                    <ul class="list-group list-group-horizontal-sm">
                        <li class="list-group-item list-group-item" style="
                        padding: 25px 0px 0px 0px; border: 0px; bottom: 10px">
                            <nav aria-label="breadcrumb"
xmlns="http://www.w3.org/1999/html">
                                <ol class="breadcrumb">
                                    <li class="breadcrumb-item active" aria-
current="page">{{ obj1.path }}</li>
                                </ol>
                            </nav>
                        </li>
                        <li class="list-group-item list-group-item"
                            style="margin: auto 0px; bottom: 10px; border:
0px; padding: 10px 0px 0px 5px">
                            <a
href="http://127.0.0.1:5000/{{obj1.slug}}?path={{ obj1.path
}}&name={{obj1.name}}&check={{ 1 }}&numbtable=1">
                                <button type="button" class="btn btn-success
btn-lg">
                                    <span>&#8592;</span>Назад
                                </button>
                            </a>
                        </li>
                    </ul>
```

```
                </div>
                {% else %}
                <div class="list-group" style="height: 75px">
                    <ul class="list-group list-group-horizontal-sm">
                        <li class="list-group-item list-group-item" style="
                    padding: 32px 0px 0px 0px; border: 0px;">
                            <nav aria-label="breadcrumb"
xmlns="http://www.w3.org/1999/html">
                                <ol class="breadcrumb">
                                    <li class="breadcrumb-item active" aria-
current="page"></li>
                                </ol>
                            </nav>
                        </li>
                        <li class="list-group-item list-group-item"
                            style="margin: auto 0px; border: 0px; padding:
10px 0px 0px 5px">
                            <a href="">
                                <button disabled type="button" class="btn
btn-success btn-lg">
                                    <span>&#8592;</span>Назад
                                </button>
                            </a>
                        </li>
                    </ul>
                </div>
                {% endif %}

                <div class="list-group" style="width: 550px;">
                    <ul class="list-group list-group-horizontal-sm">
                        <li style="width: 220px" class="list-group-
item">Ім'я</li>
                        <li style="width: 180px" class="list-group-
item">Тип</li>
                        <li style="width: 152px" class="list-group-
item">Розмір</li>
                    </ul>

                    {% for data in table1 %}
                    <div class="list-group" style="height: 40px;">
                        <a style="height: 40px" class="list-group-item list-
group-item-action"
                           href="http://127.0.0.1:5000/{{data.slug}}?path={{
data.path }}&name={{ data.name }}&check={{ 0 }}&numbtable=1">
                            <label style='width: 215px'>{{ data.name
}}</label>
                            <label style='width: 175px'>{{ data.type
}}</label>
                            <label>{{ data.size }}</label>
                        </a>
                    </div>
                    {% endfor %}
                </div>
            </div>
        </div>

        <div class="col-sm-6" style="overflow-y: scroll; height: 600px;
position: fixed; left: 750px">
            {% if obj2 != '' and obj2.name != '' %}
            <div class="list-group" style="height: 75px">
                <ul class="list-group list-group-horizontal-sm">
                    <li style="padding: 25px 0px 0px 0px; border: 0px;
bottom: 10px"
                        class="list-group-item list-group-item">
```

```
                        <nav aria-label="breadcrumb"
xmlns="http://www.w3.org/1999/html">
                            <ol class="breadcrumb">
                                <li class="breadcrumb-item active" aria-
current="page">{{ obj2.path }}</li>
                            </ol>
                        </nav>
                    </li>
                    <li class="list-group-item list-group-item"
                        style="margin: auto 0px; bottom: 10px; border: 0px;
padding: 10px 0px 0px 5px">
                        <a href="http://127.0.0.1:5000/{{obj2.slug}}?path={{
obj2.path }}&name={{obj2.name}}&check={{ 1 }}&numbtable=2">
                            <button type="button" class="btn btn-success btn-
lg">
                                <span>&#8592;</span>Назад
                            </button>
                        </a>
                    </li>
                </ul>
            </div>
            {% else %}
            <div class="list-group" style="height: 75px">
                <ul class="list-group list-group-horizontal-sm">
                    <li class="list-group-item list-group-item" style="
                    padding: 32px 0px 0px 0px; border: 0px">
                        <nav aria-label="breadcrumb"
xmlns="http://www.w3.org/1999/html">
                            <ol class="breadcrumb">
                                <li class="breadcrumb-item active" aria-
current="page"></li>
                            </ol>
                        </nav>
                    </li>
                    <li class="list-group-item list-group-item"
                        style="margin: auto 0px; border: 0px; padding: 10px
0px 0px 5px">
                        <a href="">
                            <button disabled type="button" class="btn btn-
success btn-lg">
                                <span>&#8592;</span>Назад
                            </button>
                        </a>
                    </li>
                </ul>
            </div>
            {% endif %}

            <div class="list-group" style="height: 40px; width: 550px">
                <ul class="list-group list-group-horizontal-sm">
                    <li style="width: 220px" class="list-group-
item">Ім'я</li>
                    <li style="width: 180px" class="list-group-item">Тип</li>
                    <li style="width: 152px" class="list-group-
item">Розмір</li>
                </ul>

                {% for data in table2 %}
                <div class="list-group" style="height: 40px">
                    <a style="height: 40px" class="list-group-item list-
group-item-action"
                       href="http://127.0.0.1:5000/{{data.slug}}?path={{
data.path }}&name={{ data.name }}&check={{ 0 }}&numbtable=2">
                        <label style='width: 215px'>{{ data.name }}</label>
```

```
                        <label style='width: 175px'>{{ data.type }}</label>
                        <label>{{ data.size }}</label>
                    </a>
                </div>
                {% endfor %}
            </div>
        </div>
    </div>
</div>

{% endblock %}
```